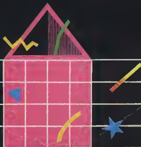
…LIN PETROVIC
…ORAN MOŠORINSKI

COMMODORE 128

PROGRAMIRANJE U BASIC-u





Mr VESELIN PETROVIĆ, dipl. mat. i ZORAN MOŠORINSKI

COMMODORE 128

PROGRAMIRANJE U BASIC-u

NIRO „TEHNIČKA KNJIGA", BEOGRAD

Mr Veselin Petrović i Zoran Mošorinski / COMMODORE 128 — Programiranje u BASIC-u

Recenzent:
dr Dušan Tošić

Urednik:
Radivoje Grbović



YU ISBN 86-325-0104-6

PREDGOVOR

Knjiga je namenjena vlasnicima Commodore-a 128, od početnika do profesionalnih programera, s tom ogradom što nema neku naročitu ambiciju da poučava već da pruži širu informaciju o jednoj mašini koja je u svetu doživela pravu eksploziju popularnosti najviše zahvaljujući svojoj potpunoj kompatibilnosti sa ranijim modelom iste firme. Ako još napomenemo da Commodore 128 pored toga što »prima« sve programe koji su pisani za C-64 poseduje veoma dobar Basic koji u potpunosti podržava sve hardverske karakteristike kompjutera koje neskromno rečeno nisu zanemarljive ni sa stanovišta mnogo naprednijih 16-bitnih mašina, upotrebna vrednost knjige je time veća jer su upravo te razlike između C-64 i C-128, a koje unapređuju raniji model do epiteta standardno, posebno obrađene.

Na početku se dosta govori o hardveru same mašine i periferijskim uređajima iz prostog razloga da shvatite šta sve sa kompjuterom možete da uradite, odnosno da sagledate sve one detalje koji čine da se C-128 smatra tako dobrim ozbiljnim računarom. U nastavku obrađene su sve Basic naredbe ali tako da su očigledne podele među njima po logičnim celinama. Ukupno je obrađeno preko 150 naredbi od kojih je većina propraćena primerom. Naravno, posebno su istaknute naredbe vezane za rad sa grafikom i zvukom jer su po svojoj obimnosti i celovitosti one te koje uzdižu ovu mašinu na jedan visok nivo.

Nadamo se da će knjiga biti shvaćena više kao priručnik jer upravo joj je i takva namena.

Zahvaljujemo se Ademu Jakupoviću na korisnim savetima koji su nam znatno pomogli u izradi ovog rukopisa.

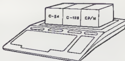
Autori

SADRŽAJ

1. UVOD

U ovom poglavlju izložene su osnovne karakteristike računara Commodore 128 (u daljem tekstu C-128), koji predstavlja nadgradnju popularnog prethodnika — računara Commodore 64 (C-64). Uslovno rečeno, C-128 se može shvatiti kao računar u kojem postoje tri računara koji koriste istu tastaturu i periferiju, a to znači da su moguća tri načina rada:

— C-64
— C-128
— CP/M



U C-64 režimu rada, računar C-128 predstavlja pravu kopiju računara C-64. Time je ostvarena potpuna kompatibilnost između računara C-64 i C-128, što nije zanemarljivo s obzirom na raznovrsnost softvera i značajan broj korisnika računara C-64.

U C-128 režimu rada dolaze do punog izražaja stvarne mogućnosti računara C-128. U njemu su eliminisani osnovni nedostaci koji karakteriču računar C-64, kao što su:

— skroman BASIC jezik
— spora disketna jedinica
— 40-to kolonski ekran.

CP/M je skraćenica za »Control Program for Microcomputer«. Reč je o operativnom sistemu smeštenom na disketi, koji je razvio *Digital*

Research, Inc CP/M režim rada namenjen je pre svega poslovnoj primeni i on omogućuje pisanje i korišćenje programa nezavisno od tipa računara.

O svakoj od pomenutih mogućnosti biće detaljnije reči u poglavlju 4 (Mogućnosti računara C-128).

Iz svega rečenog, a s obzirom na raspoložive periferijske uređaje o kojima će biti reči u poglavlju 2 (C-128 i periferijski uređaji), može se zaključiti da je C-128 kućni računar pogodan za efikasno korišćenje u edukativne svrhe, kao i za manje poslovne primene.



2 C-128 I PERIFERIJSKI UREĐAJI

Svaki računar koristi i razne priključne uređaje koji mu omogućavaju komuniciranje sa korisnikom, memorisanje podataka za duži vremenski period i povezivanje sa drugim računarima istog ili drugačijeg tipa. Ovi priključni uređaji zovu se — periferijski uređaji. Bez nekih od njih računar je apsolutno neupotrebljiv, dok drugi služe za to da računaru prošire mogućnosti. Uz računar Commodore 128 najčešće se koriste: tastatura, ekranski uređaji, disketni uređaji, štampači i modemi. Neki od periferijskih uređaja dobijaju se uz sâm računar dok se drugi moraju posebno nabaviti.

ŠTA SE DOBIJA UZ RAČUNAR?

Uz računar se dobija jedan od neophodnih periferijskih uređaja tastatura. Bez tastature ne bismo mogli računaru da saopštimo šta želimo, ne bismo mogli ni na koji način da mu prenesemo poruku ili da napišemo program. Tastatura je najuobičajenije sredstvo za unošenje podataka u računar. Kad kupite računar C-128, čini vam se da i niste dobili ništa drugo osim tastature. Međutim u plastičnom kućištu se osim tastature nalazi i sâm računar koji se sastoji od mikroprocesora 8502 (koji služi za režime rada C-64 i C-128), mikroprocesora Z80A (koji služi za rad računara pod operativnim sistemom CP/M), integrisanog kola VIC II (Video Interface Chip) koje je »zaduženo« za grafiku, integrisanog kola 8503 (koje se koristi kad računar prikazuje 80 kolona teksta), integrisanog kola SID (Sound Interface Device) koji se koristi za zvuk i dva integrisana kola CIA (Complex Interface Adaptor) za komuniciranje sa periferijskim uređajima.

U plastičnoj kutiji računara nalazi se i njegova memorija. Ona je podeljena na dva dela: ROM (Read Only Memory — memorija samo za čitanje) i RAM (Random Access Memory — memorija sa direktnim pristupom). ROM-memorija sadrži fiksno programirani operativni sistem i BASIC-interpreter, dok RAM-memoriju koristimo za sopstvene programe. C-128 ima 128 kilobajta RAM-memorije, odnosno 131072 bajta (ili znaka).

Na računaru se, osim toga, primećuju i konektori za povezivanje raznih periferijskih uređaja, o kojima će biti reči pri opisivanju pojedinih periferijskih uređaja.

EKRANSKI UREĐAJI

Ekranski uređaj je jedini uređaj koji je, osim samog računara, apsolutno neophodan da bi vaš C-128 mogao da radi. To može da bude običan televizor ili poseban uređaj koji se zove monitor. Monitor je uređaj sličan televizoru, samo što ne poseduje elektronska kola za prijem televizijskog programa. Pošto se u ovom slučaju signal ne transformiše u komplikovanim elektronskim kolima, monitor daje mnogo bolju sliku nego televizijski prijemnik. Da li ćete koristiti televizijski prijemnik ili monitor — zavisi od toga za šta želite da koristite računar.

TELEVIZIJSKI PRIJEMNIK

To je najjeftiniji ekranski uređaj, pošto skoro da i nema domaćinstva bez televizijskog prijemnika. Ovo rešenje sasvim zadovoljava ako imate nameru da računar koristite za igre ili jednostavnije programe. Glavni nedostatak televizijskog prijemnika je slabo razlaganje slike, tako da je broj detalja koji se mogu videti na ekranu prilično ograničen. Kad su u pitanju slike (kao što je to slučaj sa igrama), razlaganje televizijskog prijemnika je sasvim dovoljno, ali ne i kada je u pitanju tekst: slova na ekranu nisu oštra i zamorno ih je čitati.

Televizijski prijemnik se na računar priključuje antenskim kablom na priključak sa oznakom RF.

DIREKTNI MONITOR (U BOJI)

Direktni monitor je najjeftinija vrsta monitora u boji. Do ovog monitora treba dovesti signal za ton i dva signala za sliku (osvetljenje i boja). Slika na ovoj vrsti monitora prilično je oštra (znatno oštrija nego na televizijskom prijemniku), pa može da se koristi i za prikazivanje teksta, ali on još uvek nije najpogodniji za rad sa 80-kolonskim tekstom. Firma *Commodore* proizvodi direktni monitor sa tipskom oznakom 1702. Direktni monitor se priključuje sa zadnje strane računara na DIN-priključak sa oznakom VIDEO. Priključni kabl mora da bude predviđen za direktni monitor.

KOMPOZITNI MONITOR (U BOJI)

Kompozitni monitor koristi signal za ton i signal za sliku koji predstavlja kompozitni (ujedinjeni) signal osvetljenja i boje. Zbog toga se i zove kompozitni monitor. Njegovo razlaganje je nešto lošije nego razlaganje direktnog monitora, tako da ni on nije pogodan za prikazivanje teksta u 80 kolona. I ovaj monitor se priključuje na priključak VIDEO. Priključni kabl mora da bude predviđen za kompozitni monitor.

RGBI-MONITOR

Jedini monitor u boji na kojem se može dobiti zadovoljavajući tekst u 80 kolona je RGBI-monitor. RGBI je skraćenica od *Red-Green-Blue-Intensity* (crveno-zeleno-plavo-intenzitet). Za ovaj monitor potreban je jedan signal za ton i četiri signala za sliku: crvena, zelena, plava boja i intenzitet.

RGBI-monitor se priključuje na konektor sa oznakom RGBI na zadnjoj strani računara kablom za sliku, dok se ton dobija priključenjem na priključak VIDEO. Znači — za priključenje RGBI-monitora potrebna su dva priključna kabla.

MONOHROMNI MONITOR

Za poslovne primene u kojima obrada teksta zauzima važno mesto preporučuje se korišćenje monohromnog (crno-belog, žutog ili zelenog) monitora. Ovaj tip monitora ima najoštriju sliku i najbolji prikaz 80-kanalskog teksta, tako da najmanje zamara oči. Monohromni monitor se priključuje na priključak VIDEO.

KASETNI UREĐAJI

Kasetni uređaji služe za upisivanje programa i podataka na kasetu, kao i za njihovo čitanje sa kasete. Kasetni uređaji su najjeftiniji uređaji za memorisanje na magnetnom medijumu (magnetna traka u kaseti). Ovi uređaji koriste standardne audio-kasete za snimanje muzike i govora, tako da je i sâm magnetni medijum veoma jeftin.

Nedostaci kasetnih uređaja su mala brzina rada i sekvencijalno upisivanje podataka bez mogućnosti direktnog pristupa nekom podatku (da bi se došlo do nekog podatka upisanog na magnetnu traku, neophodno je pročitati sve podatke upisane pre njega). Kasetni uređaj se na računar C-128 priključuje odgovarajućim kablom na priključak sa oznakom CASSETTE.

DISKETNI UREĐAJI

Disketni uređaji imaju istu funkciju kao i kasetni, samo su znatno brži i omogućavaju direktan pristup podacima upisanim na magnetnom medijumu (disketi).

Uz računar C-128 može se koristiti i disketna jedinica 1541, pravljena za računar Commodore 64. Međutim, ona je prilično spora i ima mali kapacitet (170 kilobajta) za računar kao što je C-128. Zbog toga je firma *Commodore* napravila novi disketni uređaj sa tipskom oznakom 1571, koji je brži od uređaja 1451 i do deset puta (zavisno od načina rada), a kapacitet mu je 340 kilobajta. Proizvodi se i dvostruki disketni uređaj koji može da prihvati dve diskete, a tipska oznaka mu je 1572.

Disketna jedinica se na računar priključuje na priključak sa oznakom SERIAL (na zadnjoj strani računara)

ŠTAMPAČI

Štampači su uređaji koji na papiru ispisuju programe, tekstove ili rezultate obrade. Za svoje računare firma *Commodore* proizvodi četiri tipa štampača: tri matrična (MPS-801, MPS-802 i MPS-803) i jedan sa kvalitetnim otiskom (DPS-1101). Ako želite jeftin štampač i nije vam važan kvalitet slova, nabavite jedan od tri matrična modela, a ako računar želite da koristite i u poslovne svrhe (gde je potreban bolji kvalitet otiska), preporučujemo da nabavite kvalitetni matrični štampač nekog drugog proizvođača. Taj štampač treba da bude napravljen posebno za Commodore, ili je potrebno da nabavite poseban međusklop (interface).

Štampač se na računar priključuje na priključak SERIAL (isti onaj koji služi za priključenje disketnog uređaja), a ako želite da priključite i štampač i disketni uređaj — potrebno je da ih vežete redno tako da štampač bude na kraju redne veze.

MODEMI

Lični računari se sve više povezuju u mreže računara putem telefonskih linija. Uređaj koji omogućava slanje podataka telefonskom linijom zove se *modem* (što je skraćenica od modulator-demodulator). Firma *Commodore* nudi četiri tipa modema: 1600, 1650, 1660 i 1670. Brzine prenosa svih ovih modema su od 300 do 1200 boda (grubo: 30—120 znakova u sekundi). Modem 1670 ima i hardverski ugrađeno automatsko biranje telefonskog broja.

Modem se priključuje na USER PORT sa zadnje strane računara na desnoj strani (gledano sa zadnje strane).

PALICE ZA IGRU

Palice za igru su uređaji koji računaru prenose informacije o smeru (gore — dole — levo — desno) i imaju taster za okidanje, što je veoma pogodno za akcione igre, ali palica može da se koristi i uz ozbiljnije programe (ako je to programom predviđeno). U ovu grupu periferijskih uređaja spadaju i tzv. miševi koji smer određuju pomeranjem po tvrdoj podlozi. Palice za igru se priključuju na priključke CONTROL PORT 1 i CONTROL PORT 2 sa desne strane računara (do prekidača za uključenje i isključenje).

OSTALI PERIFERIJSKI UREĐAJI

Osim navedenih postoji i još čitav niz periferijskih uređaja koji mogu da se priključe na C-128. Tu spadaju crtači (ploteri), magnetne kartice, uređaji za sintezu govora, digitalizatori slike, uređaji za sintezu

muzike, protivprovalnički i protivpožarni alarmni uređaji, kao i čitav niz drugih priključnih uređaja.

Važno upozorenje: Nikad ne priključujte i ne odvajajte priključni uređaj dok je računar uključen. Takav postupak može dovesti do oštećenja periferijskog uređaja ili računara.

IZBOR KONFIGURACIJE

U zavisnosti od toga šta želite da radite na računaru, moguća je sledeća konfiguracija:

— Za igru i učenje programiranja: računar, kasetni uređaj, palica za igru (jedna ili dve) i televizijski prijemnik.

— Za obradu teksta: računar, disketni uređaj, kvalitetni štampač i monohromni monitor (ili kvalitetni monitor u boji — RGBI).

— Za manje poslovne primene: računar, kasetni uređaj, disketni uređaj, štampač (kvalitet zavisi od primene) i monohromni monitor (ili monitor u boji).

— Za veće poslovna primene: računar, kasetni uređaji, dva disketna uređaja, jedan ili dva brza štampača i monohromni monitor (ili kvalitetni monitor u boji — RGBI).

— Za povezivanje u mrežu: modem (uz periferijske uređaje prilagođene lokalnim potrebama).

3. TASTATURA

Tastatura računara C 128 sastoji se od alfanumeričkog dela, numeričkog dela i četiri grupe sa po četiri tastera. U režimu rada C-64 mogu se koristiti samo tasteri na alfanumeričkom delu i funkcionalni tasteri f1, f3, f5 i f7. U preostala dva načina rada može se koristiti kompletna tastatura.

ALFANUMERIČKI DEO

Alfanumerički deo tastature, osim tastera sa alfabetskim, numeričkim i specijalnim znacima, ima i tastere koji služe za izvršavanje određenih funkcija.

Taster SHIFT

Taster SHIFT služi za kucanje velikih slova (kad je izabran skup znakova sa malim slovima), odnosno za formiranje PET-grafike (kad je izabran skup znakova sa velikim slovima i grafikom). Za tastere koji imaju po dva znaka (gore i dole), taster SHIFT služi za dobijanje gornjeg znaka. Kad se taster SHIFT pritisne zajedno sa tasterom COMMODORE (C=), prelazi se sa jednog skupa znakova na drugi (sa skupa znakova sa malim i velikim slovima na skup znakova sa velikim slovima i PET-grafikom, ili obratno).

Taster SHIFT LOCK

Taster SHIFT LOCK ima funkciju imitiranja stalne pritisnutosti tastera SHIFT. Ako je potrebno da se otkuca veći broj znakova sa pritisnutim tasterom SHIFT, onda je pogodnije pritisnuti taster SHIFT LOCK tako da on ostane u pritisnutom položaju, otkucati željene znakove i ponovo pritisnuti taster SHIFT LOCK tako da se vrati iz pritisnutog položaja.

Taster RETURN

Taster RETURN služi za završavanje reda upisa. Njime se računaru naznačava da je završeno upisivanje reda.

Taster RESTORE

Taster RESTORE (kad se pritisne zajedno sa tasterom RUN/STOP) vraća računar na standardne parametre (tamnosivi ekran sa svetlozelenim okvirom u načinu rada C-128, odnosno plavi ekran u načinu rada C-64 kao i skup znakova sa velikim slovima i PET-grafikom)

Taster CONTROL

Ovaj taster služi za upisivanje pojedinih specijalnih znakova, kao i za akustične signale i ekranske i sistemske funkcije On se koristi uvek zajedno sa nekim drugim tasterom, tako da se pritisna taster CONTROL, a zatim (dok je on u pritisnutom položaju) i željeni taster. Taster CONTROL se često koristi za definisanje boje znakova na ekranu. Ako se pritisne taster CONTROL i jedan od tastera sa numeričkim oznakama 1—8, znak će dobiti određenu boju (moguć izbor 8 boja).

Taster RUN/STOP

Kad se ovaj taster pritisna bez tastera SHIFT, prekida se izvršenje programa pisanog u BASIC-u i računar prelazi u direktni način rada Kad se pritisne zajedno sa tasterom SHIFT izaziva učitavanje i izvršavanje prvog programa sa kasete (način rada C-64), odnosno sa diskete (način rada C-128)

Taster COMMODORE (C=)

Ovaj taster ima tri funkcije

1 zajedno sa tasterom SHIFT izaziva promenu skupa znakova (mala i velika slova ili velika slova i grafika),

2 zajedno sa numeričkim tasterima 1—8 omogućava izbor sledećih osam boja za ispitivanje znakova na ekranu,

3 u načinu rada C-128 omogućava korišćenje grafike ako je tastatura prebačena u DIN-standard

Taster CLR/HOME

Pritisak na ovaj taster (bez tastera SHIFT) izaziva vraćanje kursora u početni položaj (gornji levi ugao ekrana) Ako se taster CLR/HOME pritisne zajedno sa tasterom SHIFT, kursor se vraća u početni položaj, a ekran se briše

Taster INST/DEL

Ovaj taster služi za brisanje znakova iz teksta ili za njihovo ubacivanje u tekst Ako se koristi bez tastera SHIFT, izaziva brisanje znaka koji se nalazi neposredno ispred kursora a ako se koristi sa tasterom SHIFT, pomera tekst desno i ostavlja slobodan prostor za ubacivanje znaka.

Tasteri CRSR

Ovi tasteri služe za pomeranje kursora po ekranu (levi za pomeranje gore-dole, a desni za pomeranje levo-desno). Na tasterima je obeleženo u kom smeru se kursor pomera kad se ne koristi taster SHIFT, i u kom smeru se pomera kad se koristi taster SHIFT.

Funkcionalni tasteri

Funkcionalni tasteri smešteni su u gornjem desnom uglu tastature i imaju oznake f1, f3, f5 i f7. Ove oznake važe kad se tasteri koriste bez tastera SHIFT. Kad se tasteri koriste sa tasterom SHIFT, oznake su im f2, f4, f6 i f8. Ovi tasteri u načinu rada C-64 nemaju određenu funkciju, nego se njihova funkcija programira u programu. U načinu rada C-128 tasteri imaju unapred određene funkcije, ali se one mogu programski menjati. Unapred određene funkcije funkcionalnih tastera su:

f1 — izaziva ispisivanje naredbe GRAPHIC na trenutnom položaju kursora, tako da je potrebno upisati još samo parametre naredbe.
f2 — (taster f1 + taster SHIFT) izaziva na trenutnom položaju kursora ispisivanje naredbe DLOAD, zajedno sa početnim znakom navoda, tako da je potrebno još samo upisati ime datoteke i završni znak navoda.
f3 — izaziva izvršavanje naredbe DIRECTORY, odnosno lista sadržaj diskete koja se nalazi na kanalu 8.
f4 — (taster f3 + taster SHIFT) izaziva izvršavanje naredbe SCNCLR, odnosno briše ekran.
f5 — izaziva ispisivanje naredbe DSAVE" na trenutnom položaju kursora, i to sa početnim znakom navoda, tako da treba još samo dopisati naziv programa i završni znak navoda.
f6 — (taster f5 + taster SHIFT) izvršava naredbu RUN, odnosno izaziva izvršenje programa koji se trenutno nalazi u memoriji računara.
f7 — izaziva izvršavanje naredbe LIST, odnosno na ekranu daje listu programa koji se trenutno nalazi u memoriji.
f8 — (taster f7 + taster SHIFT) izaziva pozivanje monitora za mašinski jezik naredbom MONITOR.

Rezimirajmo sada još jednom funkcije ovih tastera:

f1 — GRAPHIC	f2 — DLOAD"
f3 — DIRECTORY (RETURN)	f4 — SCNCLR (RETURN)
f5 — DSAVE"	f6 — RUN (RETURN)
f7 — LIST (RETURN)	f8 — MONITOR (RETURN)

TASTERI ZA UPRAVLJANJE KURSOROM

Ova grupa tastera nalazi se levo od funkcionalnih tastera i služi za pomeranje kursora u željenom smeru. Za razliku od tastera CRSR,

na alfanumeričkom dělu tastature ovih tastera ima četiri i oni se koriste bez tastera SHIFT. Smer pomeranja kursora naznačen je na svakom tasteru.

TASTERI SA SPECIJALNIM FUNKCIJAMA

Ovi tasteri podeljeni su u dve grupe od po četiri tastera, a nalaze se iznad alfanumeričkog děla sa leve strane i u sredini. Ovi tasteri imaju različite funkcije koje ćemo detaljnije opisati u daljem tekstu.

Taster NO SCROLL

Ako prilikom prikazivanja na ekranu ima više podataka nego što ima mesta na ekranu, onda se podaci prikazuju tako da ekran "klizi" nagore pa se time oslobađa donji red za prikazivanje novih podataka. Tasterom NO SCROLL može se zaustaviti ovo klizanje ekrana tako da se mogu videti (i eventualno proučiti) podaci već prikazani na ekranu. Ponovno klizanje ekrana može se uspostaviti ponovljenim pritiskom na taster NO SCROLL.

Tastar 40/80 DISPLAY

Ovaj taster služi za izmenu načina prikaza teksta na ekranu (40 ili 80 kolona teksta). Ako se na ekranu prikazuje 40-kolonski tekst, pa se pritisne taster 40/80 DISPLAY — onda se prelazi na prikaz 80-kolonskog teksta, i obratno. Taster se prvim pritiskom blokira u donjem položaju, a drugim pritiskom se oslobađa (kao i tasteri SHIFT LOCK i NO SCROLL). Prebacivanja na 40, odnosno 80 kolona treba izvršiti pre uključenja ili resetovanja računara.

Taster LINE FEED

Pritiskom na ovaj taster kursor se pomera za jedan red niže i ostaje u istoj koloni.

Tastar HELP

Ako interpreter u svom radu nađe na grešku u nekom programskom redu, on javlja poruku o grešci. Pritisak na taster HELP prikazaće taj red na ekranu, i to tako da od mesta na kojem je interpreter utvrdio grešku slova budu inverzna (za 40-kolonski način rada), odnosno podvučena (za 80-kolonski način rada).

Taster ASCII/DIN

Ovim tasterom se tastatura (alfanumerički deo) prebacuje sa ASCII-standarda na DIN, i obratno. Ako je računar nabavljen u SR Nemačkoj, ovaj taster će imati i tastaturu kao na slici, dok računar proizveden u nekoj drugoj zemlji umesto ovog tastera ima još jedan taster SHIFT LOCK. Na tasterima alfanumeričkog děla tastature na levoj

strani su označeni znakovi po ASCII-standardu, a na desnoj znakovi po DIN-standardu (ako ima razlika u odnosu na ASCII-standard). I ovaj taster se pritiskom blokira u pritisnutom položaju, a sledećim pritiskom se oslobađa.

Taster ALT

Ovim tasterom se može uključiti sopstvena programski definisana tastatura, tako da se može prilagoditi bilo kom jeziku (a ne samo engleskom i nemačkom). Ovo je posebno pogodno za obradu teksta na nekom drugom jeziku (npr. srpskohrvatskom).

Taster TAB

Ovaj taster ima istu namenu kao i taster za tabulaciju kod pisaće mašine. Kad se pritisne zajedno sa tasterom SHIFT, on definiše tabulaciju na trenutnom položaju kursora. Kad se pritisne bez tastera SHIFT, izaziva pomeranje kursora na prvi sledeći položaj za koji je definisana tabulacija (tasterom TAB i tasterom SHIFT). Ako se tabulacija ne definiše, računar automatski dodeljuje tabulaciju na svakih 8 kolona.

Taster ESC

Ovim tasterom može da se izvršava čitav niz funkcija (poznatih pod imenom ESCAPE-sekvence). Za razliku od ostalih tastera koji sa nekim drugim tasterima izvršavaju posebne funkcije (kao što su tasteri SHIFT, CONTROL, COMMODORE i sl.), ovaj taster se ne pritiska zajedno sa tasterom sa kojim izvršava određenu funkciju, nego se prvo pritisne taster ESC (i otpusti), pa se onda pritisne taster koji definiše funkciju. Pomoću tastera ESC moguće je izvršavanje sledećih funkcija (ako se posle tastera ESC pritisne navedeni taster):

A — uključuje se automatsko ubacivanje znakova od trenutnog položaja kursora. Znakovi se ubacuju u tekst (ili programski red) sve dok se ne pritisne ESC-C

B — trenutni položaj kursora definiše donji desni ugao ekranskog prozora (WINDOW)

C — isključuje se automatsko ubacivanje znakova uključeno pritiskom na ESC-A

D — briše se trenutni red ekrana, a ceo ekran ispod tog reda pomera se naviše tako da popuni nastali prazan prostor na ekranu

E — isključuje treptanje kursora tako da kursor postaje mirni blok na ekranu

F — uključuje treptanje kursora koje je prethodno bilo isključeno pritiskom na ESC-E

G — dozvoljava dobijanje akustičnog signala pritiskom na tastere CONTROL-G

H — sprečava dobijanje akustičnog signala pritiskom na tastere CONTROL-G

I — iznad tekućeg reda ekrana uvodi novi red teksta, odnosno programa

J — pomera kursor na početak tekućeg reda

K — pomera kursor iza poslednjeg znaka u tekućem redu ekrana

L — dozvoljava klizanje ekrana

M — sprečava klizanje ekrana (ESC-L i ESC-M imaju istu funkciju kao i taster NO SCROLL)

N — prebacuje ekran iz inverznog načina predstavljanja znakova u normalan način predstavljanja (važi samo za rad sa 80-kolonskim ekranom)

P — briše sva znakove u tekućem redu ekrana do trenutnog položaja kursora u redu

Q — briše sve znakove u tekućem redu ekrana od trenutnog položaja kursora u redu do kraja reda

R — Prebacuje ekran sa normalnog načina prikazivanja u inverzni (važi samo za rad sa 80-kolonskim ekranom)

S — uključuje prikazivanje kursora u obliku bloka (ako je prethodno bilo uključeno prikazivanje kursora u obliku crtice pomoću ESC-U)

T — definiše gornji levi ugao ekranskog prozora (WINDOW) na trenutnom položaju kursora

U — uključuje prikazivanje kursora u obliku crtice (važi samo za 80-kolonski prikaz na ekranu)

V — pomera ceo ekran za jedan red gore.

W — Pomera ceo ekran za jedan red dole

X — prebacuje sa 80-kolonskog prikaza na ekranu na 40-kolonski prikaz, i obratno

Y — uključuje unapred definisane tabulacije (na svakih 8 kolona)

Z — briše sve postavljene tabulacije

@ — briše ekran od trenutnog položaja kursora do kraja

DODATNE FUNKCIJE TASTERA CONTROL

Osim funkcija koje smo već naveli, taster CONTROL u načinu rada C-128 ima i dodatne funkcije koje se postižu kad se taster CONTROL pritisne zajedno sa nekim drugim tasterom. Te funkcije (u zavisnosti od tastera koji se pritisne uz taster CONTROL) su sledeće

B — uključuje podvlačenje svih znakova koji se dalje kucaju (svi znakovi do povratka na normalni način prikazivanja biće podvučeni — važi samo za 80-kolonski prikaz)

G — izaziva akustični signal (zvonce)

I — Izaziva skok kursora na sledeću definisanu tabulaciju

J — izvršava pomeranje za jedan red.

Tastatura

K — sprečava prebacivanje sa jednog skupa znakova na drugi (sa skupa sa malim i velikim slovima — na skup sa velikim slovima i grafikom, i obratno)

L — omogućava prebacivanje sa jednog na drugi skup znakova

O — uključuje treptanje svih znakova koji se dalje prikazuju (sve do povratka na normalan način prikazivanja)

X — postavlja ili briše tabulaciju.

[— zamenjuje pritisak na taster ESC.

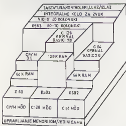
Napomena Sve funkcije koje se postižu tasterima ESC i CONTROL mogu se programirati i izvršavati iz programa.

NUMERICKA TASTATURA

Numerička tastatura nalazi se na desnoj strani i sadrži cifre od 0 do 9 znakove + i —, tačku i taster ENTER Svi ovi tasteri imaju istu funkciju kao i tasteri na aflanumeričkom delu tastature a služe za lakše unošenje numeričkih podataka Taster ENTER ima istu funkciju kao i taster RETURN

4. MOGUĆNOSTI RAČUNARA C-128

Kao što je već rečeno u poglavlju 1, na računaru C-128 moguća su tri režima rada C-64, C-128 i CP/M. U ovom odeljku biće više reči o svakom od njih. Uz to treba pomenuti da računar C-128 raspolaže RAM memorijom od 128K, sa mogućnošću proširenja do 512K.



C-64

Ovaj način rada interesantan je onim korisnicima koji su prethodno imali računar C 64, jer pruža mogućnost da se na računaru C-128 radi u istom ambijentu kao u slučaju računara C-64 (na isti način, sa istim mogućnostima i softverom). Ovo je svakako značajna po-

godnost s obzirom na raznovrstan softver koji postoji (mnogobrojni programski jezici, programi za obradu teksta, rad sa tabelama, rad sa bazom podataka, programi finansijske prirode, edukativni programi, programi za razne tipove simulacija, rad sa grafikom i muzikom, uslužni programi, širok spektar raznih tipova igara i sl.).

Prelaz u C-64 način rada može se realizovati komandom GO64 ili pritiskom na COMMODORE taster u trenutku uključivanja računara. Pri tome ne postoji komanda koja obezbeđuje povratak iz C-64 u C-128 način rada. Jedini način da se to ostvari je isključivanje i ponovno uključivanje računara.

Konfiguracija potrebna za ovakav način rada zavisi pre svega od potreba korisnika i softvera koji on želi da koristi. U najvećem broju slučajeva ovu konfiguraciju pored računara čine

— TV aparat,
— jedinica kasete,
— palica za igru,

pri čemu u nekoj boljoj konfiguraciji TV aparat treba zameniti 40-to-kolonskim monitorom, a jedinicu kasete jedinicom diskete. Uz sve to po potrebi može se pridodati štampač.

Osnovni jezik u C-64 režimu rada je BASIC 2.0, čije su mogućnosti skromne sa aspekta kućnih računara i koje znatno zaostaju za mogućnostima samog hardvera. Ne postoje naredbe za realizaciju grafike, muzike, sprajtova, kao i naredbe koje omogućuju strukturno programiranje.

Pored nekonformnog BASIC jezika, nedostatak svakako predstavlja i prilično spora disketna jedinica (posebno u slučaju korišćenja većih datoteka i dužih programa). U ovom režimu rada ograničenje predstavlja i dužina linije koja se može prikazivati na ekranu (najviše 40 simbola), što nije beznačajno u slučaju bilo koje ozbiljnije primene.

Iz svega rečenog može se zaključiti da ovaj način rada ne bi trebalo da bude posebno interesantan za one čitaoce koji nisu ranije koristili računar C-64, bez obzira na raspoloživi softver. Što se tiče korisnika računara C-64, oni bi svakako u nekom razumnom vremenu trebalo da se orijentišu na jedan od dva preostala načina rada s obzirom na pogodnosti koje oni pružaju.

C-128

U ovom režimu rada računar C-128 koristi isti hardver kao i u slučaju C-64 načina rada i podržava periferiju koja je bila dostupna računaru C-64. Razlike koje postoje mogu se posmatrati iz aspekta

a) hardvera

— dvostruko veća memorija u odnosu na računar C-64,
— mogućnost korišćenja 80-to kolonskog ekrana,
— poboljšana tastatura,
— znatno brže disketne jedinice,

b) softvera:

— izmene u KERNAL operativnom sistemu,
— izmene u ekranskom editoru
— ugrađen mašinski monitor,
— poboljšan BASIC jezik proširenjem verzije 2.0 u 7.0,
— nov način formatizacije diskete (disketne jedinice 1571 i/ili 1572)

U odnosu na C-64, računar C-128 ima dvostruko veću raspoloživu RAM memoriju (128K) pri čemu postoji mogućnost proširivanja ove memorije do 512K dodavanjem novih memorijskih modula.

Kada je reč o ekranu, moguće je koristiti 40-to kolonski ili 80-to kolonski ekran, što se može odabrati odgovarajućim tasterom. Pri tome 80-to kolonski ekran zahteva monitor određenog tipa.

Tastatura računara je značajno izmenjena. Postojećim tasterima je pridodata izdvojena grupa numeričkih tastera, kao i izvestan broj novih tastera, kao što su: ESC, TAB, HELP, 40/80 DISPLAY, LINE FEED ...

Problem spore disketne jedinice 1541 na računaru C-64 prevaziđeo je korišćenjem novih disketnih jedinica:

— Commodore 1571
— Commodore 1572

U računar C-128 ugrađen je monitor-program koji omogućuje pisanje asemblerskih programa, disasembliranje i testiranje asemblerskih programa, kao i ispitivanje i modifikacije sadržaja memorije.

Nova verzija BASIC jezika (BASIC 7.0), koja je implementirana u računaru C-128, predstavlja značajno proširenje BASIC jezika (BASIC 2.0) računara C-64. Ove izmene odnose se na naredbe za rad sa

— grafikom
— muzikom i
— sprajtovima,

kao i na naredbe koje omogućuju strukturno programiranje. Takođe postoje naredbe/komande koje olakšavaju pripremu i testiranje programa (AUTO, RENUMBER, TRACE, HELP ...)

Značajno poboljšanje predstavlja i naredba GRAPHIC koja omogućuje definisanje jednog od šest grafičkih mödova (način prikazivanja informacija na ekranu). Jednim od njih moguće je definisati i tzv. "deljenje ekrana" (na grafički i tekstualni deo), što je veoma korisno u određenim slučajevima.

Novinu predstavlja, kao što je već rečeno u poglavlju 3, da su funkcionalni tasteri F1, F2, ..., F8 unapred definisani i da imaju određeno značenje. U BASIC-u 7.0 postoji komanda KEY kojom je moguće promeniti neku od ovih definicija, shodno potrebama u programu.

Što se tiče same konfiguracije, ona može biti, kao što je rečeno u poglavlju 2, veoma raznolika i zavisi pre svega od materijalnih mogućnosti i potreba korisnika.

S obzirom na pomenuto, C-128 će svakako biti najinteresantniji način rada za većinu korisnika, te ćemo u ovoj knjizi računar C-128 uglavnom i posmatrati sa tog aspekta.

CP/M

Korisnici računara C-64, koji su imali na raspolaganju disketnu jedinicu 1541, svakako su upoznali komande BASIC jezika koje omogućuju korišćenje pomenute disketne jedinice. One su bile neophodne pre svega za upisivanje nekog programa na disketu ili učitavanja programa sa diskete u unutrašnju memoriju računara, kao i za neke druge funkcije.

Adekvatne komande postoje i u BASIC-u računara C-128. Štaviše skup raspoloživih komandi je proširen. Ove komande su svakako neophodne i u slučaju drugih programskih jezika.

Iz prethodnog se može zaključiti da disketna jedinica zahteva softver koji bi omogućio njeno korišćenje.

Godine 1975. po prvi put je realizovan softverski paket CP/M (operativni sistem za mikroračunare koji obezbeđuje kompletnu kontrolu rada sa disketnom jedinicom). Time je omogućeno da korišćenje programa pisanih u režimu CP/M operativnog sistema bude nezavisno od tipa računara. Na ovaj način znatno je proširen raspoloživi softver za računar C-128.

S obzirom na pomenutu činjenicu, CP/M način rada je svakako najprihvatljiviji ukoliko računar treba koristiti u poslovne svrhe. Ograničavajući faktor u izvesnom smislu predstavlja hardver, čija cena nije zanemarljiva.

Za razliku od C-64 i C-128 režima rada, kod kojih je operativni sistem rezidentno smešten u ROM-u koji se neposredno nakon uključivanja računara automatski prebacuje u RAM, u CP/M načinu rada operativni sistem je smešten na disketi u obliku programa koji se učitava i aktivira nakon uključivanja računara — ako je CP/M sistemska disketa u disketnoj jedinici.

U prva dva slučaja (C-64 i C-128 režim rada) koristi se mikroprocesor 8502, za razliku od CP/M režima rada u kome se koristi mikroprocesor Z80, ugrađen u računar C-128.

Za rad u CP/M režimu neophodna je disketna jedinica, monitor (40-to ili 80-to kolonski) i tzv. 'sistemske diskete' (diskete na kojima je smešten CP/M softver) koje se dobijaju uz računar. Zavisno od potreba moguće je priključiti neki od štampača, a eventualno i modem.

Budući da je ovaj način rada, s obzirom na hardver koji on zahteva, neprihvatljiv za većinu korisnika, to se u ovoj knjizi nećemo ni baviti ovim načinom rada.

5. OSNOVE BASIC-JEZIKA

5.1. UVODNI POJMOVI

Jedna od komponenti sistemskog softvera je programski jezik tipa "interpreter", ugrađen u ROM-u računara sa ciljem da obezbedi njegovo normalno funkcionisanje. Pomenuti programski jezik karakteriša osobina da se svaka naredba programa najpre interpretira, a zatim izvršava. Stoga se programi pisani u njemu izvršavaju relativno sporo ali uglavnom brzinom koja ipak zadovoljava. Njegova dobra strana ogleda se u tome što je moguće veoma efikasno izvršavati eventualne izmene u programu.

U najvećem broju slučajeva BASIC jezik računara C-128 predstavlja zadovoljavajući programski jezik za rešavanje problema. U daljem izlaganju koristićemo sledeće osnovne pojmove:

— simboli BASIC jezika,
— podatak,
— promenljiva,
— izraz,
— niz.

Simbole BASIC jezika čine simboli koji su upisani na samoj tastaturi računara (velika i mala slova engleske azbuke, cifre dekadnog brojnog sistema, specijalni znaci, grafički simboli, simbol "blanko", simboli koje definiše korisnik) i koji predstavljaju azbuku za formiranje novih elemenata jezika.

Podatak predstavlja zapisanu informaciju koja je predmet obrade na računaru. Moguća su dva tipa podataka: *brojni* i *znakovni*. Brojni podatak predstavlja numeričku vrednost izraženu u dekadnom brojnom sistemu, pri čemu može biti uključen i znak. Brojni podatak može biti u pozicionom ili eksponencijalnom zapisu.

Znakovni podatak predstavlja linijski niz simbola i mora biti omeđen znacima navoda. Broj simbola u znakovnom podatku određuje tzv. "dužinu podataka".

Pod *promenljivom* podrazumevamo simboličku oznaku kojoj se mora dodeliti podatak. S obzirom na prirodu podatka, promenljiva može biti *brojna* ili *znakovna*. Svaku promenljivu karakterišu dva parametra: naziv promenljive i tekuća vrednost (podatak koji je dode-

ljen promenljivoj). Naziv promenljive može biti proizvoljne dužine, pri čemu su značajna samo prva dva simbola u nazivu. Prvi simbol obavezno mora biti slovo, a ostali mogu biti slova ili brojevi.

Znakovna promenljiva se koristi za imenovanje niza simbola BASIC jezika, dužine do 255 karaktera. Ova promenljiva obavezno kao poslednji simbol u nazivu promenljive ima simbol $

Brojnoj promenljivoj se može dodeliti isključivo brojni podatak. Ova promenljiva može biti *celobrojna* ili *realna* promenljiva. Celobrojna promenljiva obavezno ima simbol % kao poslednji simbol u nazivu promenljive. Vrednost koja se može dodeliti ovoj promenljivoj mora biti iz opsega [—32768,32767]

Svaka promenljiva je po internoj konvenciji realna (ukoliko nije eksplicitno naznačena kao znakovna ili celobrojna). Vrednosti koje se mogu dodeliti ovoj promenljivoj moraju biti reda veličine [10^{-38}, 10^{38}]

Zapis postupka za dobijanje jednog rezultujućeg podatka na osnovu jednog ili više zadatih podataka naziva se *izraz*. On može biti *brojni* ili *znakovni* — u zavisnosti od toga da li je rezultat brojna ili znakovna vrednost

U brojnim izrazima dozvoljene su operacije

— sabiranja (+)
— oduzimanje (—)
— množenje (*)
— deljenje (/)
— stepenovanje (↑)
— promena znaka (—)

U znakovnim izrazima dozvoljena je operacija

— dopisivanje (+)

U okviru izraza (brojnog i/ili znakovnog) mogu se koristiti operacije poređenja

— jednakost (=)
— nejednakost (<>)
— manje od (<)
— manje ili jednako (<=)
— veće od (>)
— veće ili jednako (>=)

kao i logičke operacije

— i (AND)
— ili (OR)
— ekskluzivno ili (XOR)
— ne (NOT)

Ako izraz sadrži više od jedne operacije tada se vrednost izraza određuje tako što se operacije izvršavaju sleva nadesno, vodeći ra-

čuna o prioritetu operacija. Utvrđeni prioritet operacija može biti promenjen korišćenjem zagrada. U narednoj tabeli dat je prioritet operacija, pri čemu je sa 1 naznačen najveći prioritet.

Operacije	Prioritet
Promena znaka	1
Stepenovanje	2
Množenje, deljenje	3
Sabiranje, oduzimanje	4
Operacije poređenja	5
NOT	6
AND	7
OR, XOR	8

Skup podataka sa zajedničkim imenom naziva se *niz*. Pojam niza se uvodi da bi se izbeglo pojedinačno imenovanje podataka u slučajevima kada se u programu pojavljuje veći skup ulaznih podataka, međurezultata ili izlaznih podataka.

Nizovi mogu biti sa jednom ili više dimenzija. Mogu se koristiti za brojne ili znakovne podatke. Pri tom važi ista notacija kao i u slučaju promenljivih. Indeksiranje uvek počinje sa 0.

U C-64 režimu rada na raspolaganju je verzija 2.0 BASIC jezika koja je identična sa BASIC jezikom na računaru Commodore 64.

U C-128 režimu rada BASIC 2.0 zamenjen je BASIC-om koji predstavlja značajno proširenja u odnosu na prethodni. Pridodat je veći broj novih naredbi za rad sa grafikom, sprajtovima, muzikom i periferijskim jedinicama, za deljenje ekrana na tzv. "prozore" i strukturno programiranje. Uveden je izvestan broj novih funkcija, a proširen je i skup komandi koje služe za efikasnije obrađivanje programa. Pri tom je BASIC 2.0 pravi podskup BASIC-a 7.0 (sve funkcije, naredbe i komande iz verzije 2.0 važe i u verziji 7.0).

U BASIC-u 7.0, ukoliko se koristi 40-to kolonski ekran, tada na raspolaganju stoje isti setovi simbola kao u slučaju računara C-64 (set velikih i malih slova ili set velikih slova i grafičkih simbola). S druge strane, za slučaj 80-to kolonskog ekrana moguće je istovremeno korišćenje velikih i malih slova, kao i grafičkih simbola.

Bez obzira na način rada o kome je reč (C-64 ili C-128), računar se može koristiti dvojako:

— u direktnom načinu rada,

— u odloženom načinu rada.

U *direktnom načinu rada* računar predstavlja neku vrstu kalkulatora i pisaće mašine. U ovom slučaju naredba se izvršava odmah nakon što se unese i pritisne taster RETURN. Pri tome instrukcije nije sačuvana u memoriji računara.

U *odloženom načinu rada* naredbe se unose jedna za drugom i smeštaju se u memoriju računara. Grupa unetih naredbi se pamti i predstavlja program koji kasnije može biti izvršen. U daljem izlaganju uglavnom ćemo se baviti ovim načinom rada.

Ovde ćemo izložiti osnovne naredbe BASIC jezika, grupisane u određene logičke celine. Pri tome neće biti eksplicitne podele na BASIC 2.0 i BASIC 7.0 već će pri navođenju same naredbe, komande ili funkcije biti naznačeno o kojoj verziji BASIC jezika je reč. Pre toga reći ćemo nešto o strukturi samih naredbi. Naime, svaka naredba sadrži u sebi ključnu reč koja se nalazi na početku naredbe i koja definiše njenu osnovnu namenu.

Na početku naredbe se može nalaziti broj iz intervala [1,63999] koji je jedinstveno određen u okviru naredbi jednog programa i koji služi za definisanje pozicije naredbe u okviru programa. Pomenuta osobina omogućuje umetanja nove naredbe u okviru postojećeg skupa naredbi. Nakon svakog kompletiranja naredbe potrebno je pritisnuti taster RETURN. Naredbe se smeštaju u memoriju, a njihovo izvršenje se aktivira pomoću komande RUN. Ako je broj na početku naredbe izostavljen, tada se naredba izvršava odmah nakon njenog unošenja.

U jednom redu može biti sadržano više naredbi, pri čemu su one obavezno razdvojene simbolom "dve tačke". Dužina jedne linije programa može biti 160 znakova (u C-128 načinu rada), odnosno 80 simbola (u C-64 načinu rada).

U daljem izlaganju koristićemo sledeću notaciju:

zapis velikim slovima — definiše ključnu reč BASIC jezika,

zapis malim slovima — ukazuje da je sadržaj promenljivog karaktera i definiše ga korisnik konkretnim sadržajem.

[ ] — simbol koji ukazuje da sadržaj u okviru zagrade može biti izostavljen,

{ } — simbol koji ukazuje da treba odabrati tačno jedan element naveden u okviru zagrade.

( ) — ovaj simbol predstavlja sastavni deo sintakse same naredbe.

5.2. NAREDBE ULAZA/IZLAZA I NJIMA PRIDRUŽENE NAREDBE

Ove naredbe omogućuju prihvatanje podataka sa tastature, disketne jedinice ili jedinice trake, kao i njihovo dodeljivanje promenljivim, odnosno ispisivanje podataka (i/ili programa) na ekranu monitora/TV aparata, disketi, traci ili štampaču.

Ovoj grupi pripadaju naredbe: INPUT, GET, GETKEY, PRINT, PRINT USING, PUDEF, DATA, READ, RESTORE, CLR, WINDOW, OPEN, INPUT#, GET#, PRINT#, CMD, DOPEN, CLOSE, DCLOSE, DCLEAR, APEND i RECORD.

INPUT (Basic 2.0)

Služi za prihvatanje podataka sa tastature i njihovo dodeljivanje ulaznim veličinama. Opšti oblik naredbe je

INPUT ["izraz";] lista

pri čemu je

izraz — znakovni izraz koji se prikazuje na ekranu u trenutku kada se očekuje prihvatanje podatka/podataka sa tastature računara (opcionog je karaktera i služi kao olakšica pri izvršavanju programa),

lista — jedna ili više promenljivih, međusobno razdvojenih zarezima, kojima se dodeljuju vrednosti prihvaćene sa tastature, ;

— simbol koji služi za razdvajanje znakovnog izraza od promenljivih naznačenih u listi

Naredba se izvršava na taj način što se privremeno prekida izvršenje programa, na ekranu se prikaže znakovni izraz naznačen u naredbi INPUT (ukoliko je naveden), iza njega se prikaže znak ? i očekuje se unos vrednosti promenljivih naznačenih u listi. Vrednosti se unose jedna za drugom međusobno razdvojene zarezima, pri čemu na kraju treba pritisnuti taster RETURN. Potrebno je da postoji usaglašenost po broju i tipu promenljivih naznačenih u listi i vrednosti koje se unose.

Ova naredba se može koristiti isključivo unutar programa.

Napomena: Dužina podataka u C-64 môdu je maksimalno 80 karaktera, a u C-128 môdu 160 karaktera.

GET (Basic 2.0)

Koristi se za prihvatanje jednog znaka (karaktera) određenim trenutnim delovanjem na tastaturu. Opšti oblik naredbe je:

GET lista

pri čemu je:

lista — jedan ili više argumenata (najviše 10), međusobno razdvojenih zarezima, od kojih se svakom pridružuje po jedan karakter sa tastature. Argumentima naznačenim u listi inicijalno se dodeljuje vrednost nula — ukoliko se radi o brojnim, odnosno "blanko" — ukoliko se radi o znakovnim promenljivim.

Nakon upisivanja karaktera koji će biti prihvaćen/prihvaćeni naredbom GET nije potrebno pritisnuti taster RETURN. Prihvaćeni karakteri se smeštaju u posebnu zonu memorije koja se naziva *bafer tastature*. Naredba GET se može koristiti isključivo na nivou programa.

Napomena: Pri izvršavanju naredbe GET program ne je u stanju čekanja da se izvrši neka akcija na tastaturi. U tu svrhu potrebno je realizovati tzv. "petlju".

GETKEY (Basic 7.0)

Ima istu namenu kao i naredba GET. Razlika je u tome što je pri izvršenju naredbe GETKEY računar u stanju čekanja sve dok korisnik ne otkuca neki karakter na tastaturi. Opšti oblik naredbe je

GETKEY lista

pri čemu parametar **lista** ima isto značenje kao u slučaju naredbe GET.

PRINT (Basic 2.0)

Koristi se za ispisivanje brojnih vrednosti i/ili teksta na ekranu monitora/TV aparata. Opšti oblik naredbe je

PRINT [lista]

pri čemu je.

lista — spisak elemenata koji mogu biti podaci, promenljive, brojni izrazi, znakovni podaci, tekst omeđen znacima navoda. Elementi liste mogu među sobom biti razdvojeni zarezom, tačkazarezom ili simbolom blanko, što je od značaja za uređivanje odnosa između podataka/teksta naznačenih u listi pri njihovom ispisivanju na ekran.

Naredba se izvršava na taj način što se na ekranu ispisuju vrednosti elemenata naznačenih u listi i to na sledeći način

— ukoliko je lista izostavljena, tada se na ekranu ispisuje jedan prazan red,

— ukoliko je elemenat brojna ili znakovna promenljiva, tada se na ekranu ispisuje tekuća vrednost promenljive,

— ukoliko je elemenat liste brojni ili znakovni izraz, tada se najpre određuje vrednost izraza a zatim se na ekranu ispisuje njegova vrednost,

— ukoliko je elemenat liste tekst naznačen između znakova navoda tada se na ekranu ispisuje taj tekst bez znakova navoda

U okviru teksta koji se prikazuje na ekranu mogu se koristiti i karakteri koji imaju specijalno značenje. Njima se može obezbediti pomeranje kursora, rad sa bojama, upotreba rezervnih karaktera (izmena boje osnove i karaktera koji se po njoj ispisuju) i setovanje velikih/malih slova

Svi specijalni karakteri, sa izuzetkom RETURN i INST/DEL tastera, ukoliko su navedeni u okviru znaka navoda, imaju tzv 'odloženo dejstvo', tj. aktiviraju se tek pri izvršavanju odgovarajuće PRINT naredbe U toku unošenja ovi karakteri imaju poseban izgled na ekranu.

Za pomeranje kursora po ekranu koriste se tasteri CLR/HOME, SHIFT CLR/HOME ⇑CRSR⇓, SHIFT ⇑ CRSR ⇓ ⇐ CRSR ⇒, SHIFT ⇐CRSR⇒, ↑, ↓, ←, →

Reverzni karakteri aktiviraju se istovremenim pritiskom tastera *CONTROL* i *9*, nakon čega treba uneti tekst koji će biti prikazan u

reverznom zapisu Povratak u normalni način prikazivanja realizuje se istovremenim pritiskom tastera *CONTROL* i *O*

Aktiviranje jedne od 16 boja, raspoloživih na C-128, realizuje se istovremenim pritiskom tastera *CONTROL* i *cifra*, odnosno *COMMODORE* i *cifra* (pri tomme je 1 ≤ *cifra* ≤ 8)

PRINT naredba se može koristiti bilo u direktnom načinu rada bilo na nivou programa

PRINT USING (Basic 7.0)

Koristi se za ispisivanje brojnih i/ili znakovnih podataka na ekranu monitora/TV aparata, po zadatom formatu Opšti oblik naredbe je

PRINT USING format , lista

pri čemu je

format — parametar omeđen znacima navoda koji služi za definisanje formata podatka koji će biti ispisan na ekranu. Kao elementi formata mogu se koristiti simboli

(#) — simbol koji služi za definisanje pozicije jednog karaktera u izlaznom polju (može se koristiti pri definiciji bilo brojnog bilo znakovnog podatka),

(+) — simbol koji se može naznačiti u prvoj ili poslednjoj poziciji polja formata da bi se ispisao znak podatka, i to znak + ukoliko je brojna vrednost pozitivna (bez obzira na to da li je označena), odnosno znak — ukoliko je brojna vrednost negativna,

(—) — simbol koji se može naznačiti u prvoj ili poslednjoj poziciji polja formata da bi se ispisao znak podatka ukoliko je on negativan, u protivnom odgovarajuća pozicija ostaje prazna,

(.) — simbol koji ukazuje na položaj decimalne tačke u okviru numeričkog podatka koji će biti prikazan

(,) — simbol koji se može koristiti pri opisu formata numeričkih podataka i služi za smeštanje zareza u okviru numeričkog polja koje se prikazuje,

($) — simbol koji ukazuje da znak dolar ($) treba upisati ispred numeričkog podatka,

(↑↑↑↑) — simbol kojim se ukazuje da numerička podatak treba prikazati u eksponencijalnom zapisu (tzv E+ format),

(=) — simbol koji se koristi pri definiciji formata znakovnog podatka kada je potrebno izvršiti centriranje podatka u okviru polja formata,

(>) — simbol koji se koristi pri definiciji formata znakovnog podatka u slučaju da je potrebno izvršiti desno poravnanje

Parametar lista ima isto značenje kao u slučaju naredbe PRINT Naredba PRINT USING se može koristiti bilo u direktnom načinu rada, bilo na nivou programa

PUDEF (Basic 7.0)

Ova naredba se koristi u kombinaciji sa naredbom PRINT USING. Omogućuje da se u okviru podatka čiji je prikaz obezbeđen naredbom PRINT USING izvrši zamena svih pojava karaktera.

— blanko i/ili
— zarez i/ili
— decimalna tačka i/ili
— znak dolar

sa proizvoljno odabranim znacima. Opšti oblik naredbe je

PUDEFF "abcd"

pri čemu

a — predstavlja simbol kojim će biti zamenjene sve pojave simbola *blanko*

b — predstavlja simbol kojim će biti zamenjene sve pojave simbola *zarez*,

c — predstavlja simbol kojim će biti zamenjene sve pojave simbola *decimalna tačka*,

d — predstavlja simbol kojim će biti zamenjene sve pojave simbola *dolar*

Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

DATA (Basic 2.0)

Koristi se za smeštanje podataka u okviru programa koji će biti pročitani naredbom READ. Opšti oblik naredbe je

DATA lista

pri čemu je *lista* spisak elemenata među sobom razdvojenih zarezima. Elementi liste mogu biti brojni i/ili znakovni podaci. Znakovni podaci ne moraju biti omeđeni znacima navoda, osim ako u sebi ne sadrže simbol zarez.

Ova naredba nije izvršna, već je pridružena nekoj READ naredbi. Najčešće je smeštena na kraju programa. Mora postojati usaglašenost između podataka u naredbi DATA i imena promenljivih naznačenih u odgovarajućoj READ naredbi (reč je o usaglašenosti po tipu).

Naredba DATA se ne može koristiti u direktnom načinu rada.

READ (Basic 2.0)

Koristi se za učitavanje podataka (naznačenih u DATA naredbi) koji se dodeljuju promenljivim naznačenim u naredbi READ. Opšti oblik naredbe je

READ lista

3*

pri čemu je *lista* jedna ili više brojnih, znakovnih ili indeksnih promenljivih međusobno razdvojenih zarezima.

Pri izvršavanju ove naredbe sekvencijalno (redom) se čitaju vrednosti podataka naznačenih u jednoj (ili više) DATA naredbi i te se vrednosti dodeljuju promenljivim u READ naredbi. Ova naredba je slična INPUT naredbi s tim što su podaci koji se učitavaju fiksni i unapred definisani naredbom DATA u okviru programa. Mora postojati usaglašenost po vrsti između podataka naznačenih u naredbi DATA i promenljivih u odgovarajućoj naredbi READ.

Jedna READ naredba može pristupati jednoj ili više DATA naredbi i to u redosledu njihovog navođenja. Postoji i mogućnost pridruživanja više READ naredbi jednoj istoj DATA naredbi.

Ova naredba se može koristiti bilo u direktnom načinu rada bilo na nivou programa.

RESTORE (Basic 2.0)

Koristi se za inicijalizaciju podataka naznačenih u naredbi DATA. Opšti oblik naredbe je

RESTORE [broj]

pri čemu je

broj — obeležje DATA naredbe počev od koje se tretira njena inicijalizacija. Ukoliko se izostavi, podrazumeva se da je početna pozicija naredbe RESTORE prva naredba u programu.

Ova naredba ima sledeću funkciju: svaki put kada se izvrši READ naredba, pokazivač koji ukazuje na određenu DATA naredbu menja se i dobija vrednost koja ukazuje na sledeću DATA naredbu. Pomoću naredbe RESTORE u svakom trenutku se može modifikovati pomenuti pokazivač, tako da ukaže na naredbu čiji je broj jednak ili veći od veličine broja naznačenog u naredbi RESTORE.

Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Napomena. Naredba RESTORE u BASIC-u 2.0, na računaru C-64, ima opšti oblik RESTORE, tj. nije dozvoljeno korišćenje parametra *broj*.

CLR (Basic 2.0)

Koristi se za oslobađanje RAM memorije koja je bila dodeljena podacima, pri čemu BASIC program, smešten u memoriji računara ostaje nepromenjen. Opšti oblik naredbe je

CLR

Nakon aktiviranja CLR naredbe računar "zaboravlja" tekuće vrednosti promenljivih i nizova promenljivih, uključujući i brojače za FOR/NEXT petlje (o kojima će kasnije biti reči) kao i pokazivač na naredbu DATA. U ovom slučaju se takođe gube adrese koje obezbeđuju povratak iz potprograma.

Napomena. U slučaju rada sa datotekama pre izvršavanja naredbe CLR potrebno je zatvoriti sve datoteke (vidi CLOSE naredbu, koja će biti obrađena kasnije), bilo da se nalaze na traci, disketi ili modemu jer će u protivnom biti oštećene. Naime, nakon naredbe CLR gube se brojevi datoteka, te nije moguće njihovo kasnije zatvaranje.

WINDOW (Basic 7.0)

Koristi se za definisanje određenog dela ekrana kao radnog područja (tzv. "prozora"). Opšti oblik naredbe je

WINDOW x1, y1, x2, y2 [, broj]

pri čemu je:

x1 — celobrojna vrednost koja definiše kolonu gornjeg levog ugla prozora ($0 \leq x1 \leq 39$ ili $0 \leq x1 \leq 79$),

y1 — celobrojna vrednost koja definiše vrstu gornjeg levog ugla prozora ($0 \leq y1 \leq 24$),

x2 — celobrojna vrednost koja definiše kolonu donjeg desnog ugla prozora ($0 \leq x2 \leq 39$ ili $0 \leq x2 \leq 79$),

y2 — celobrojna vrednost koja definiše vrstu donjeg desnog ugla prozora ($0 \leq y2 \leq 24$)

broj — parametar koji ukazuje da li pri kreiranju prozora treba obrisati ekran ili ne. Moguće su vrednosti

0 — nema brisanja ekrana (podrazumevana vrednost),

1 — brisanje ekrana.

Nakon aktiviranja određenog prozora — on postaje radno područje koje služi bilo za prihvatanje ili ispisivanje podataka (i/ili programa). Naredba WINDOW se može koristiti bilo u direktnom načinu rada bilo na nivou programa.

Napomene

a) Moguće vrednosti parametara *x1* i *x2* zavise od toga da li je reč o 40-to ili 80-to kolonskom ekranu.

b) Postoji još jedan način za kreiranje prozora ekrana. U tu svrhu služe *ESC-T* i *ESC-B*, respektivno za određivanje gornjeg levog i donjeg ugla ekrana (vidi poglavlje 3).

OPEN (Basic 2.0)

Koristi se za otvaranje kanala (komunikacione veze između računara i određene periferijske jedinice) za ulaznu i/ili izlaznu jedinicu. Opšti oblik naredbe je

OPEN broj [, jedinica] [, adresa] { [, "naziv [, tip, način]"] / , "komanda" }

pri čemu je:

broj — celobrojna vrednost koja se dodeljuje datoteci pri otvaranju. Može biti iz opsega 1—255. Ovaj parametar se pojavljuje u INPUT#, GET#, PRINT# i CLOSE naredbi da bi ukazao na odgovarajuću datoteku.

jedinica — parametar koji ukazuje na eksternu jedinicu na kojoj se nalazi datoteka koja se otvara. Ovaj parametar može biti

0 — tastatura,
1 — jedinica trake (podrazumevana vrednost),
2 — RS232 jedinica (najčešće modem),
3 — ekran,
4—7 — štampač(i),
8—15 — jedinica diskete/jedinice disketa.

adresa — parametar koji definiše operaciju koju treba izvršiti u zavisnosti od jedinice. Osnovne vrednosti parametara su

Jedinica	Adresa	Značenje
Jedinica trake	0	čitanje trake
	1	zapis na traku
	2	zapis na traku sa oznakom kraja trake
Disketna jedinica	1—14	otvaranje kanala za podatke
	15	otvaranje kanala za komande
Štampač	0	velika slova/grafički simboli
	7	velika/mala slova

naziv — parametar koji definiše naziv datoteke, dužine do 16 karaktera (obavezan je za jedinicu diskete)

tip — definiše tip datoteke (PROGRAM, SEQUENTIAL, RELATIVE, USER), ovaj parametar se odnosi isključivo na jedinicu diskete,

način — definiše način rada sa datotekom: čitanje (R) ili pisanje (W); odnosi se isključivo na jedinicu diskete

komanda — neka od postojećih komandi za jedinicu trake ili jedinicu diskete

Ova naredba se može koristiti bilo u direktnom načinu rada bilo na nivou programa

INPUT# (Basic 2.0)

Ova naredba je slična naredbi INPUT, s tim što se podaci ne prihvataju sa tastature, već iz datoteke sa eksterne jedinice (trake ili diskete) koja je prethodno otvorena za čitanje. Opšti oblik naredbe je

INPUT#broj,lista

pri čemu je

broj — celobrojna vrednost dodeljena datoteci pri otvaranju (vidi OPEN naredbu),

lista — jedan ili više argumenata međusobno razdvojenih zarezima, kojima se dodeljuju vrednosti prihvaćene sa specificirane jedi-

nice. Pri tome se vrednosti koje se učitavaju moraju slagati po broju i tipu sa argumentima u listi.

Naredba INPUT# se može koristiti isključivo na nivou programa.

Napomene

a) U C-128 modu, ulazni podaci koji se učitavaju sa eksterne jedinice mogu biti dužine do 160 karaktera (159 znakova i kôd za RETURN, tj. CHR$(13)).

b) Naredbi INPUT# mora obavezno prethoditi odgovarajuća naredba OPEN.

GET# (Basic 2.0)

Ova naredba je slična GET naredbi, s tim što se znak ne prihvata sa tastature, već iz datoteke sa specificirane eksterne jedinice (kaseta ili disketa). Opšti oblik naredbe je

GET#broj,lista

pri čemu parametar *broj* predstavlja celobrojnu vrednost koja se dodeljuje datoteci pri otvaranju, a parametar *lista* ima isto značenje kao u slučaju naredbe GET. Ova naredba se može koristiti isključivo na nivou programa.

PRINT# (Basic 2.0)

Ova naredba je slična naredbi PRINT, s tim što se brojne vrednosti i/ili tekst ne ispisuju po ekranu, već u datoteci na specificiranoj jedinici (koja u krajnjoj instanci može biti ekran). Pri tome pomenuta datoteka mora biti prethodno otvorena za pisanje. Opšti oblik naredbe je

PRINT#broj,lista

pri čemu je

broj — celobrojna vrednost dodeljena datoteci pri otvaranju (vidi OPEN naredbu),

lista — spisak elemenata koji mogu biti brojni podaci, brojni izrazi, znakovni podaci ili tekst pod znacima navoda. Elementi se mogu razdvajati zarezom, tačka-zarezom. Sintaksa navođenja elemenata u listu zavisi od jedinice na kojoj se vrši zapis.

Naredba PRINT# se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

CMD (Basic 2.0)

Monitor/TV ekran služi kao standardna izlazna jedinica. CMD naredba omogućuje promenu izlazne jedinice, tj. izlazni rezultati se mogu

uputiti u datoteku na disketi, traci ili štampaču (koja se prethodno mora otvoriti). Opšti oblik naredbe je

CMD broj [, "tekst"]

pri čemu je

broj — celobrojna vrednost dodeljena datoteci pri otvaranju (vidi OPEN naredbu),

tekst — znakovni podatak koji će biti ispisan na izlazu pre nego što se izvrši naredba koja sledi naredbu CMD. Opcionog je karaktera.

CMD naredba se može aktivirati bilo u direktnom načinu rada bilo na nivou programa.

Napomena. CMD naredba se najčešće koristi u kombinaciji sa PRINT ili LIST naredbom.

DOPEN (Basic 7.0)

Naredba slična naredbi OPEN, s tim što se odnosi isključivo na disketnu jedinicu. Služi za otvaranje datoteke (programske, sekvencijalne ili relativne) na disketi radi njenog čitanja i/ili pisanja. Postoje dva zapisa ove naredbe

DOPEN # broj, "naziv [, tip]" [, Ddisk, Ujedinica] [, W]

ili

DOPEN # broj, "naziv" [, Ldužina] [, W]

Prvi oblik se odnosi na programsku i sekvencijalnu datoteku, a drugi isključivo na relativnu datoteku, pri čemu parametri imaju sledeće značenje

broj — logički broj datoteke koji predstavlja celobrojnu vrednost koja se dodeljuje datoteci pri otvaranju i služi za njenu identifikaciju u daljem radu (INPUT#, PRINT#, GET#, CLOSE, DCLOSE). Može biti iz opsega 1—255, pri čemu brojevi veći od 127 imaju specijalno značenje,

naziv — parametar koji definiše naziv datoteke (dužine do 16 karaktera). Ukoliko se kao parametar unese znakovna promenljiva, tada ona mora biti naznačena unutar zagrada. Ovaj parametar određuje naziv programske ili sekvencijalne datoteke,

tip — parametar za definiciju tipa datoteke (programske, sekvencijalne). Moguće vrednosti su

P — za programske datoteke, tj. programe,
S — za sekvencijalne datoteke.

Parametar je opcionog tipa i ukoliko se ne navede — podrazumeva se sekvencijalna datoteka (ukoliko je reč o kreiranju datoteke). Za slučaj operacije čitanja otvara se postojeća datoteka sa naznačenim imenom, bez obzira na njen tip (programska ili sekvencijalna),

disk — parametar kojim se definiše logička vrednost dodeljena disketnoj jedinici. Moguće vrednosti su 0 i 1. Parametar je opcionog tipa i ukoliko se izostavi podrazumeva se *D0*,

jedinica — parametar kojim se definiše fizička vrednost dodeljena disketnoj jedinici. Moguće vrednosti su 8 i 9. Parametar je opcionog tipa i ukoliko se izostavi podrazumeva se *U8*,

W — parametar kojim se definiše otvaranje datoteke za operaciju pisanja. Opcionog je karaktera. U slučaju programske i sekvencijalne datoteke (prvi oblik naredbe DOPEN), parametar mora biti naznačen ako treba vršiti upisivanje u datoteku, a u protivnom se podrazumeva operacija čitanja. Za slučaj relativne datoteke (drugi oblik naredbe DOPEN), parametar može biti uvek izostavljen, jer se relativna datoteka uvek otvara i za operaciju čitanja i za operaciju pisanja,

naziv — parametar koji definiše naziv relativne datoteke (dužine do 16 karaktera),

dužina — ovaj parametar se odnosi isključivo na relativnu datoteku i definiše dužinu sloga. Maksimalna vrednost parametra je 254 karaktera. Parametar je obavezan u slučaju otvaranja datoteke za operaciju kreiranja i opisivanja novog sloga u datoteku.

Simboli D, U i L, koji se pojavljuju u naredbi DOPEN, sastavni su deo sintakse i obavezni su ukoliko je naveden odgovarajući parametar.

Naredba DOPEN se može koristiti bilo u direktnom načinu rada bilo na nivou programa.

CLOSE (Basic 2.0)

Koristi se za zatvaranje datoteke ili kanala jedinice trake/diskete/štampača/modema. Opšti oblik naredbe je

CLOSE broj

pri čemu je:

broj — celobrojna vrednost dodeljena datoteci ili jedinici (traka, disketa, štampač, modem) pri otvaranju (vidi naredbu OPEN).

Napomena. Svaka datoteka koja se otvori (naredba OPEN i DOPEN) mora se i zatvoriti.

Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

DCLOSE (Basic 7.0)

Naredba slična naredbi CLOSE, s tim što se odnosi isključivo na disketnu jedinicu. Služi za zatvaranje jedne ili svih datoteka na disketi koje su prethodno otvorene. Opšti oblik naredbe je

DCLOSE [#broj] [{ON / ,} Ujedinica]

pri čemu je

broj — logički broj datoteke koji je dodeljen datoteci pri otvaranju (vidi naredbu DOPEN) Parametar je opcionog tipa, i ukoliko se izostavi — podrazumeva se zatvaranje svih datoteka koje su trenutno otvorene

Parametar *jedinica* ima isto značenje kao u slučaju naredbe DOPEN

Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa

DCLEAR (Basic 7.0)

Koristi se za zatvaranje svih otvorenih kanala i datoteka koje se odnose na određenu disketnu jedinicu Opšti oblik naredbe je

DCLEAR [Ddisk [ON] Ujedinica]

pri čemu parametri *disk* i *jedinica* imaju isto značenje kao u slučaju naredbe DOPEN

Naredba DCLEAR se može koristiti bilo u direktnom načinu rada, bilo na nivou programa

APPEND (Basic 7.0)

Služi za otvaranje sekvencijalne datoteke u koju treba dopisati nove podatke na kraju datoteke tj za „proširivanje" datoteke Opšti oblik naredbe je

APPEND#broj 'naziv' [Disk [ON] Ujedinica]

pri čemu parametri *broj naziv*, *disk* i *jedinica* imaju isto značenje kao u slučaju naredbe DOPEN

Pri izvršenju naredbe APPEND otvara se datoteka i pokazivač se postavlja na kraj datoteke, što omogućuje da se nakon toga naredbom PRINT# izvrši upis novih podataka u datoteku

Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa

RECORD (Basic 7.0)

Koristi se za pozicioniranje pokazivača na određenu lokaciju u relativnoj datoteci Opšti oblik naredbe je

RECORD#broj slog [, karakter]

pri čemu je

broj — logički broj koji je dodeljen datoteci pri otvaranju (vidi naredbu OPEN ili DOPEN),

slog — celobrojna vrednost koja predstavlja redni broj sloga u relativnoj datoteci. Ovaj parametar može biti iz opsega [0,65535],

karakter — celobrojna vrednost koja predstavlja željenu poziciju karaktera u okviru naznačenog sloga. Ovaj parametar može biti iz opsega [1,254]

PRIMERI

Primer 1 — Iskoristimo računar za sračunavanje:

a) 237+8
b) (421—9)*7
c) (2^5—12)/5

Upišimo najpre naredbu

```
PRINT 237+8
```

a zatim pritisnemo taster RETURN. Odmah nakon toga u sledećem redu ekrana pojaviće se rezultat 245.

Na sličan način možemo realizovati i sračunavanja izraza pod b) i c), pri čemu su odgovarajuće naredbe

```
PRINT (421—9)*7
```

odnosno

```
PRINT (2 ↑ 5—12)/5
```

koje respektivno daju rezultate 2884 i 4.

U ovom primeru koristili smo računar u direktnom načinu rada. Naime, odmah posle upisivanja naredbe i pritiska na taster RETURN naredba biva izvršena, a pri tom njen sadržaj nije sačuvan u memoriji računara (budući da smo izostavljali broj na početku naredbe).

Primer 2 — Upišimo sada sledeće programske redove

```
10 INPUT "VASE IME ",I$
20 INPUT "ADRESA ",A$
40 PRINT "VASE IME JE: ";I$
50 PRINT "VASA ADRESA: ";A$
```

Nakon unošenja svake od naznačenih naredbi treba pritisnuti taster RETURN, čime se ukazuje da je naredba kompletna. S obzirom na to da pomenute naredbe sadrže obeležja (brojeve linija) one se smeštaju u memoriju računara, i to u rastućem redosledu obeležja.

Da bismo aktivirali program, potrebno je upisati RUN a zatim pritisnuti taster RETURN. Nakon toga, u sledećem redu ekrana (u odnosu na tekuću poziciju kursora), pojavljuje se poruka

```
VASE IME ?
```

Od nas se očekuje da upišemo ime recimo ZORAN, i tada pritisnemo taster RETURN U sledećem redu se prikazuje poruka

ADRESA ?

Upišmo na primer, NEMANJINA 17 KRALJEVO i pritisnemo RETURN Na ekranu ce se, nakon dva prazna reda, prikazati

VASE IME JE ZORAN
VASA ADRESA NEMANJINA 17 KRALJEVO

Napomena Pre nego što pređemo na sledeći primer, da bismo izbrisali prethodni sadržaj memorije, upišmo

NEW

a zatim pritisnamo taster RETURN O komandi NEW biće kasnije reči Za sada upamtimo ovo kao postupak koji treba ponoviti pre razmatranja bilo kog novog primera

Primer 3 — Program ilustruje razliku između naredbi GETKEY i GET

```
10 PRINT PRITISNITE BILO KOJI TASTER
20 GETKEY A$
30 PRINT 'PRITISNUT JE TASTER ' A$
```

Nakon aktiviranja programa na ekranu se prikazuje poruka

PRITISNITE BILO KOJI TASTER

Od nas se očekuje određena akcija — kako je to naznačeno u poruci Pritisnimo taster W (pri tom ne treba pritisnuti taster RETURN) Nakon toga na ekranu će se prikazati poruka

PRITISNUT JE TASTER W
READY

Poruka READY ukazuje da je program kompletno izvršen

Ponavljanjem postupka izvršenja programa više puta može se uočiti da naredba GETKEY A$ u programskoj liniji 20 zadržava dalje izvršenje programa sve dok se ne pritisne neki taster (bez obzira na vreme koje protekne od trenutka kada se na ekranu pojavi poruka PRITISNITE BILO KOJI TASTER pa sve dok se ne pritisne neki taster)

Izmenimo prethodni program na taj način što umesto programske linije 20 upišemo novi sadržaj

```
20 GET A$
```

Ovo možemo realizovati na taj način što jednostavno upišemo prethodnu liniju, a zatim pritisnemo taster RETURN. Pri izvršavanju novodobijenog programa, čiji je zapis

```
10 PRINT "PRITISNITE BILO KOJI TASTER"
20 GET A$
30 PRINT "PRITISNUT JE TASTER " A$
```

možemo uočiti da se odmah nakon poruke PRITISNITE BILO KOJI TASTER na ekranu pojavljuje i poruka

```
PRITISNUT JE TASTER
READY
```

što ukazuje da naredba GET A$ nije zadržala dalje izvršenje programa već se odmah nakon njenog izvršenja (u trenutku kada se ona izvršavala, mi zbog brzine kojom se naredbe izvršavaju nismo uspeli da pritisnemo nijedan taster) prešlo na izvršavanje sledeće naredbe koja ujedno označava i kraj programa.

Ovaj nedostatak se može prevazići o čemu će kasnije još biti reči. U ovom trenutku je značajno uočiti razliku između pomenute dve naredbe (GET i GETKEY).

Primer 4 — Ilustrovanje nekih mogućnosti naredbe PRINT (korišćenje karaktera sa specijalnim značenjem u okviru teksta koji se prikazuje).

U sledećem programu koristićemo notaciju

(SHIFT/CLR HOME) — simbolički zapis da treba istovremeno pritisnuti tastere SHIFT i CLR HOME

(CONTROL/9) — simbolički zapis da treba istovremeno pritisnuti tastere CONTROL i 9

Analogno važi i za (CONTROL/0), (CONTROL/3) i (CONTROL/2).

```
10 PRINT "(SHIFT/CLR HOME)"
20 PRINT "(CONTROL/9) PRITISNITE DVA TASTERA (CONTROL/0)"
30 GETKEY A$, B$
40 PRINT "(CONTROL/3) PRITISNUTI SU TASTERI (CONTROL/2)" A$ B$
```

Nakon aktiviranja programa ekran će biti obrisan, a u vrhu ekrana biće ispisan tekst

PRITISNITE DVA TASTERA

u reverznom zapisu. Pritisnimo, na primer, tastere 1 i X. Nakon toga na ekranu će se pojaviti tekst:

```
PRITISNUTI SU TASTERI: 1        X
```

Napomena. Iz prethodnog primera možemo zapaziti da se naredbom GETKEY može prihvatiti više znakova sa tastature.

Primeri 5, 6 i 7 ilustruju korišćenje naredbe PRINT USING za slučaj znakovnih podataka.

Primer 5 — Program ilustruje korišćenje naredbe PRINT USING (simbol #)

```
10 INPUT "UPISI ZNAKOVNI PODATAK: ",A$
20 PRINT USING "##",A$
30 PRINT USING "####",A$
40 PRINT USING "#######",A$
50 PRINT USING "##########",A$
```

Aktivirajmo program i upišimo BEOGRAD kao znakovni podatak. Nakon pritiska tastera RETURN na ekranu će se pojaviti:

```
BE
BEOG
BEOGRAD
BEOGRAD
```

Iz prethodnog primera se može zaključiti da broj simbola # u polju formata definiše dužinu izlaznog polja. U slučaju da je polje formata kraće od dužine podatka koji treba prikazati, tada se vrši odsecanje podatka (sdesna) shodno veličini polja formata.

Primer 6 — Program ilustruje korišćenje naredbe PRINT USING (simbol =)

```
10 INPUT "A$ ",A$
20 INPUT "B$ ",A$
30 INPUT "C$ ",C$
40 PRINT "123456789": PRINT
50 PRINT USING "=########",A$
60 PRINT USING "=########",B$
70 PRINT USING "=########",C$
```

Aktivirajmo program i upišimo kao ulazne veličine respektivno MILAN, ANA, NEMANJA. Na ekranu se dobija izlaz:

```
123456789
  MILAN
   ANA
 NEMANJA
```

Primer 7 — Ilustruje korišćenje naredbe PRINT USING (simbol >)

Program je sličan prethodnom, s tim što je simbol (=) zamenjen simbolom (>)

Nakon izvršenja programa ekran ima izgled

```
123456789
     MILAN
       ANA
   NEMANJA
```

Primeri 8 i 9 ilustruju korišćenje naredbe PRINT USING za slučaj brojnih podataka

Primer 8 — Program ilustruje korišćenje naredbe PRINT USING (simboli #, ., ,, +, —)

```
10 INPUT "A=",A
20 INPUT "B=",B
30 PRINT
40 PRINT USING "#######",A,B
50 PRINT USING "#####.#",A,B
60 PRINT USING "###,###",A,B
70 PRINT USING "####.#+",A,B
80 PRINT USING "####.#-",A,B
```

Nakon aktiviranja programa upišimo kao ulazne veličine A=—5273 i B=235.64 Kao rezultat dobijamo·

```
  —5273      236
 —5273.0    235.6
  —5,273     236
 5273.0—   235.6+
 5273.0—    235.6
```

Ponovo aktivirajmo program upisujući kao ulazne veličine A=—23459 i B=32.49 Kao rezultat dobijamo

```
 —23459     32
 *******   32.5
 —23,249    32
 *******  32.5+
 *******   32.5
```

Iz prethodnog primera se mogu izvesti dva zaključka

1) Pre prikazivanja numeričkog podatka po naznačenom formatu obavlja se zaokruživanje podataka shodno opisu formata, a zatim se vrši njegovo prikazivanje

2) Ukoliko je polje formata nedovoljno da prihvati podatak koji treba prikazati, tada se čitavo polje popunjava simbolom *

Primer 9 — Program ilustruje korišćenje naredbe PRINT USING (simboli $ i ↑↑↑↑)

```
10 INPUT "A= ";A
20 PRINT
30 PRINT USING "+######";A
40 PRINT USING "+###.##";A
50 PRINT USING "$####.#";A
60 PRINT USING "#$###.#";A
70 PRINT USING "###.#↑↑↑↑";A
```

Za ulazni podatak A=58.46 dobijaju se respektivno rezultati

```
+     58
+  58.46
    58.5
  $58.5
5.85E+01
```

Napomene

1) Naredbe sa obeležjem 50 i 60 ilustruju korišćenje simbola $ na fiksnoj poziciji, odnosno na tzv. "plivajućoj" poziciji.

2) Naredba sa obeležjem 70 ilustruje korišćenje tzv. eksponencijalnog zapisa brojnog podatka.

Primer 10 — Korišćenje naredbe PUDEF. Aktiviranjem programa

```
20 A=3786.392
30 PRINT USING "###,###.##";A
```

dobijamo izlaz oblika

```
3,786.39
```

Izvršimo izmenu u programu dodajući sledeću programsku liniju

```
10 PUDEF "*.,"
```

kojom se zahteva zamena

— *blanko* karaktera karakterom *zvezdica*,
— simbola *zarez* simbolom *decimalna tačka*,
— simbola *decimalna tačka* simbolom *zarez*.

S obzirom na prethodno, izvršavanjem programa dobijamo izlaz oblika

```
**3.786,39
```

Primer 11 — Program ilustruje korišćenje naredbi DATA i READ

```
10 READ A,B,X$,C%,Y$
20 PRINT "A,B,C = ";A,B,C%
30 PRINT "X$,Y$ = ";X$,Y$
40 DATA 11.37,43.9,JUGOSLAVIJA,65,"CAO, KLINCI"
```

Pri aktiviranju programa najpre se izvršava naredba READ tj realnim promenljivim A i B se redom dodeljuju vrednosti 11.37 i 43.9 znakovnoj promenljivoj X$ vrednost JUGOSLAVIJA, celobrojnoj promenljivoj C% vrednost 65, a znakovnoj promenljivoj Y$ vrednost CAO, KLINCI. S obzirom na to da vrednost dodeljene poslednjoj promenljivoj sadrži u sebi simbol *zarez*, ona mora biti navedena pod znacima navoda.

Nakon toga naredbama sa obeležjima 20 i 30 vrši se prikazivanje sadržaja pomenutih promenljivih.

Primer 12 — Program ilustruje naredbe RESTORE

```
10 READ A,B: PRINT A,B
20 RESTORE 60
30 READ X,Y,Z: PRINT X,Y,Z
40 DATA 1,2
50 DATA NENAD,PREDRAG,MLADEN
60 DATA 666,777,888
```

Pri izvršavanju programa najpre se numeričkim promenljivim A i B dodeljuju respektivno brojne vrednosti 1 i 2, definisane naredbom DATA u liniji 40, a zatim se na ekranu prikazuju vrednosti promenljivih A i B, tj. ekran dobija izgled

```
1          2
```

Nakon toga prelazi se na izvršavanje naredbe RESTORE 60, tj. obavlja se inicijalizacija podataka definisanih naredbom DATA u liniji 60. To znači da se interni pokazivač postavlja na prvi podatak u pomenutoj DATA naredbi (tj. ukazuje na vrednost 666). Dalje se učitavaju vrednosti 666,777,888 i one se dodeljuju redom promenljivim X, Y i Z. Nakon prikazivanja ovih vrednosti ekran ima izgled

```
1    2
666  777  888
```

Primer 13 — Program ilustruje korišćenje naredbe CLR. Izvršimo izmenu u programu navednom u primeru 11, dodajući sledeću programsku liniju.

```
50 PRINT A,B,C%
```

Aktiviranjem programa zaključujemo da je sadržaj promenljivih A B i C% ostao nepromenjen (naredbe sa obeležjima 20 i 50 daju isti rezultat). U oba slučaja biće A=11.37, B=43.9, C=65.

Izvršimo sada još jednu izmenu. Naime, dopišimo naredbu.

```
45 CLR
```

i ponovo aktivirajmo program. U ovom slučaju naredbe sa obeležjima 20 i 50 prikazuju različite vrednosti promenljivih A, B i C%, jer su naredbom CLR pomenute promenljive resetovane na inicijalne vrednosti. Pri tome će naredba sa obeležjem 20 dati A=11.37, B=43.9 i C%=65, a naredba sa obeležjem 50 A=0, B=0 i C%=0.

Primer 14 — Korišćenje naredbi OPEN, PRINT# i CLOSE

```
10 OPEN 2,8,2,"PODACI,S,W"
20 PRINT#2,"SRBIJA"
30 PRINT#2,"HRVATSKA"
40 PRINT#2,"SLOVENIJA"
50 CLOSE 2
```

Ovim primerom prikazan je najelmentarniji način upisivanja podataka u sekvencijalnu datoteku. Pod datotekom podrazumevamo skup podataka koji predstavlja jedinstvenu celinu. U slučaju sekvencijalne datoteke podaci se upisuju redom jedan za drugim.

Prva informacija u programu služi za otvaranje kanala 2 (prvi parametar u naredbi OPEN) koji se dodeljuje datoteci. Eksterna jedinica na kojoj se nalazi datoteka, u ovom primeru disketa, određena je drugim parametrom u naredbi OPEN. Sledeći parametar u istoj naredbi definiše kanal za prenos podataka. Poslednji parametar određuje naziv datoteke (PODACI), tip (sekvencijalna) i operaciju (upis).

Nakon otvaranja datoteke, naredbama sa obeležjima 20, 30 i 40 vrši se upisivanje podataka SRBIJA, HRVATSKA i SLOVENIJA u datoteku PODACI, posredstvom kanala 2.

Naredba za zatvaranje datoteke određena je programskim redom 50.

Na analogan način se može napisati program za čitanje prethodno formirane datoteke. Budući da datoteku treba čitati, programski red 10 treba izmeniti u

```
10 OPEN 2,8,2,"PODACI,S,R"
```

S druge strane, naredbe PRINT#1 u programskim redovima 20, 30 i 40 treba zameniti naredbama INPUT#1, i to respektivno:

```
20 INPUT#2,A$
30 INPUT#2,B$
40 INPUT#2,C$
```

Ukoliko bismo želeli da proverimo pročitane podatke, potrebno je dopisati i novi programski red

```
60 PRINT A$,B$,C$
```

Primer 15 — Korišćenje naredbi DOPEN, DCLOSE i DCLEAR

```
10 DOPEN#1,"PODACI",W
20 PRINT#1,"SRBIJA"
30 PRINT#1,"HRVATSKA"
40 PRINT#1,"SLOVENIJA"
50 DCLOSE#1
60 DCLEAR
```

Ovaj program ima istu namenu kao i program u primeru 14, s tim što su u ovom slučaju korišćene naredbe BASIC-a 7.0 (DOPEN, DCLOSE i DCLEAR) koje se odnose isključivo na disketnu jedinicu.

Primer 16 — Korišćenje naredbe APPEND

```
20 APPEND#1,"PODACI",D0 ON U8
30 PRINT#1,"VOJVODINA"
40 DCLOSE#1
```

Primer 17 — Korišćenje naredbe RECORD

Kao što je ranije rečeno, naredba RECORD se koristi isključivo u radu sa relativnim datotekama. U ovom trenutku pomenimo osnovne karakteristike relativne datoteke, kako bi program koji ćemo izložiti bio što razumljiviji.

U primeru 14 susreli smo se sa pojmom datoteke kao skupom podataka. Pri tom je datoteka izdeljena u grupe podataka (slogove) koji čine određene logičke celine. Elementi slogova nazivaju se polja.

Osnovna razlika između sekvencijalne datoteke (koju smo koristili u pomenutom primeru) i relativne datoteke sastoji se u tome što je određenom slogu sekvencijalne datoteke moguće pristupiti tek nakon što se pristupi svim slogovima koji mu prethode, dok je kod relativne datoteke moguć direktan pristup. Štaviše, moguć je direktan pristup na određenu poziciju u okviru sloga. Ova osobina relativne datoteke omogućena je njenom posebnom organizacijom, tako da se slogu pristupa preko indeksa koji je određen relativnom pozicijom sloga u odnosu na početak datoteke.

Vratimo se sada zadatku koji želimo da rešimo. Pretpostavimo da treba formirati relativnu datoteku sa dva sloga čiji su sadržaji

JELENA, NOVI SAD
DRAGAN, SARAJEVO

pri čemu su redni brojevi tih slogova u datoteci redom 5 i 2. Odgovarajući program biće

```
10 DOPEN#1,"RELATIVNA",L20
20 RECORD#1,5,1
30 PRINT#1,"JELENA","NOVI SAD"
40 RECORD#1,2,1
50 PRINT#1,"DRAGAN","SARAJEVO"
60 DCLOSE
```

Primer 18 — Korišćenje naredbe CMD

Pretpostavimo da treba prethodni program izlistati na štampaču. U tom cilju ostavićemo prethodni program u memoriji računara, a u direktnom načinu rada redom upisati naredbe

```
OPEN 5,4
CMD 5
LIST
PRINT#5
CLOSE 5
```

Prva naredba služi za otvaranje datoteke (kojoj je pridružen broj 5) na štampaču (eksterna jedinica definisana vrednošću 4). Naredba CMD služi za promenu standardne izlazne jedinice (ekrana) u datoteku na štampaču. Komanda LIST, o kojoj će kasnije biti reči, služi za prikaz programskih linija koje se nalaze u memoriji računara. PRINT naredba se koristi za pražnjenje kanala koji je dodeljen štampaču naredbom CMD. Poslednja naredba obezbeđuje zatvaranje datoteke.

5.3 NAREDBE/KOMANDE KOJE SE KORISTE PRI PISANJU, KORIŠĆENJU I IZMENAMA U PROGRAMU

U ovom poglavlju biće obrađene naredbe: REM, STOP, END, LET, WAIT, DEF FN, DIM i SLEEP kao i komande AUTO, LIST, DELETE, RENUMBER, KEY, CONT, NEW i RUN.

Napomena: Generalno uzev, osnovna razlika između naredbi i komandi sastoji se u tome što su naredbe sastavni deo BASIC jezika i što one predstavljaju osnovu za pisanje programa. Za razliku od njih, komande su namenjene samom radu sa računarom, pre svega kao pomoć pri pisanju programa, i one obično ne učestvuju u programima. U slučaju kućnih računara ova podela nije tako oštra, pa je većina komandi dostupna i na nivou programa, o čemu će biti reči u svakom konkretnom slučaju.

REM (Basic 2.0)

Koristi se za postavljanja linija komentara u okviru programa. Opšti oblik naredbe je:

REM tekst

pri čemu *tekst* može biti ma koji sadržaj i predstavlja komentar.

REM naredba ukazuje računaru da sadržaj koji se nalazi s desne strane reči REM pa sve do pritiska na taster RETURN predstavlja komentar koji nema značaja pri izvršavanju programa. U okviru programa može postojati proizvoljan broj REM linija u bilo kom delu programa.

Ova naredba se može koristiti bilo u direktnom, bilo u odloženom načinu rada.

STOP (Basic 2.0)

Koristi se za prekid daljeg izvršenja programa. U jednom programu može postojati više STOP naredbi. Opšti oblik naredbe je

STOP

Ova naredba je pogodna za pronalaženje grešaka u programu. Njome je moguće zaustaviti izvršenje programa, nakon čega se može proveriti sadržaj određenih promenljivih a zatim komandom CONT nastaviti dalje izvršenje programa.

Aktiviranjem naredbe STOP računar prikazuje poruku

BREAK IN broj—linije

Ova naredba se može aktivirati bilo u direktnom načinu rada, bilo na nivou programa.

END (Basic 2.0)

Služi za zaustavljanje izvršenja programa. Nakon izvršenja ove naredbe na ekranu se pojavljuje poruka READY, koja ukazuje da je računar spreman za dalje korišćenje. Opšti oblik naredbe je

END

Ova naredba se može zadati bilo u direktnom načinu rada, bilo na nivou programa.

LET (Basic 2.0)

Koristi se za izračunavanje izraza, brojnog ili znakovnog, i za dodeljivanje izračunate vrednosti brojnoj/znakovnoj promenljivoj. Opšti oblik naredbe je

[LET] promenljiva = izraz

pri čemu je:

promenljiva — promenljiva koja može biti brojnog ili znakovnog tipa, shodno tipu izraza koji se nalazi s desne strane znaka jednakosti;

izraz — parametar koji definiše izraz koji može biti brojnog ili znakovnog tipa.

Službena reč LET može biti izostavljena. Ova naredba se može zadati bilo u direktnom načinu rada, bilo na nivou programa.

WAIT (Basic 2.0)

Služi za privremeno zaustavljanje programa — sve dok se ne ostvari određeni uslov, tj. dok određena memorijska lokacija ne dobije određenu vrednost. Opšti oblik naredbe je

```
WAIT lokacija, m1 [,m2]
```

pri čemu je

lokacija — parametar kojim se definiše memorijska lokacija čiji se sadržaj koristi u WAIT naredbi. Uglavnom se koriste lokacije koje se odnose na hardver i čije se vrednosti menjaju nezavisno od toka BASIC programa,

m1, m2 — numerički izrazi koji se prevode u celobrojne vrednosti.

WAIT naredba se najčešće koristi za privremeno zaustavljanje programa uz slanje odgovarajuće poruke (u ulazno/izlaznim aktivnostima). Ona se može koristiti za komunikaciju BASIC programa sa nekim delom hardvera.

Naredba se izvršava na sledeći način: najpre se izvršava AND logička operacija, pri čemu su operandi sadržaj memorijske lokacije definisane parametrom *lokacija* i *m1*. Nakon toga izvršava se logička operacija ekskluzivno OR, pri čemu su operandi rezultat AND operacije i *m2*. Ako dobijeni rezultat ima vrednost 0, tada se ponavlja prethodni postupak. U protivnom — nastavlja se sa izvršavanjem programa.

WAIT naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

DEF FN (Basic 2.0)

Služi za definisanje numeričke funkcije koja će kasnije biti korišćena u okviru programa. Opšti oblik naredbe je

```
DEF FN naziv(promenljiva) = izraz
```

pri čemu je

naziv — parametar koji definiše ime koje je dodeljeno novodefinisanoj funkciji. Prvi karakter u nazivu mora biti slovo, a ostali — slova ili cifre. Naziv može biti proizvoljne dužine, pri čemu su bitna prva dva simbola,

promenljiva — predstavlja numeričku promenljivu koja služi kao argument funkcije,

izraz — bilo koji numerički izraz.

DEF FN naredba omogućuje definisanje komplikovanih numeričkih izraza kao funkcija, što predstavlja pogodnost ako takve izraze treba pozivati na više mesta u programu.

U okviru programa, novodefinisana funkcija se poziva sa

FN naziv(promenljiva)

pri čemu je argumentu *promenljiva* dodeljena konkretna vrednost. Ova naredba se može koristiti isključivo na nivou programa.

DIM (Basic 2.0)

Omogućuje dodeljivanje jedinstvenog imena grupi podataka i rezervaciju memorijskog prostora u računaru za pomenutu grupu podataka. Ova naredba se koristi za brojne/znakovne podatke. Opšti oblik naredbe je

DIM ime($n_1,n_2,\ldots,n_k$)

pri čemu je

ime — parametar koji definiše ime i tip (brojni, znakovni, celobrojni) k-dimenzionalnog niza

$n_1,\ldots,n_k$ — parametri koji određuju dimenzije k-dimenzionalnog niza i broj elemenata u svakoj dimenziji (počev od 0)

Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

SLEEP (Basic 7.0)

Omogućuje zaustavljanje izvršenja programa određeni broj sekundi. Opšti oblik naredbe je

SLEEP broj

pri čemu je

broj — brojna vrednost iz intervala [0,65535], koja definiše vreme (izraženo u sekundama) za koje treba izvršiti privremeni prekid programa.

Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

AUTO (Basic 7.0)

Koristi se za automatsko numerisanje programskih linija pri upisivanju programa. Opšti oblik komande je

AUTO [broj]

pri čemu je

broj — celobrojna vrednost iz opsega [1,63999] koja predstavlja *korak* između dve uzastopne programske linije

Nakon aktiviranja komande AUTO (pri čemu je naveden i parametar *broj*) potrebno je prvi programski red uneti kompletno (sa obeležjem programskog reda na početku), a zatim pritisnuti taster RETURN. U sledećem redu ekrana automatski će se ispisati obeležje sledećeg programskog reda (početna vrednost uvećana za korak). Potrebno je samo upisati sadržaj programskog reda i pritisnuti taster RETURN. Ova osobina je posebno pogodna pri pisanju većih programa.

Prekid daljeg programa realizuje se pritiskom na taster RETURN odmah nakon što se na ekranu pojavi brojna vrednost koja predstavlja obeležje.

Poništavanje dejstva komande *AUTO broj* realizuje se sa:

AUTO

Komanda AUTO se može koristiti isključivo u direktnom načinu rada.

LIST (Basic 2.0)

Služi za prikazivanje (listanje) jedne ili više linija BASIC programa koji se nalazi u memoriji računara. Opšti oblik komande je:

```
      ⎧ blanko      ⎫
      ⎪ broj1—broj2 ⎪
LIST  ⎨ broj1—      ⎬
      ⎪ —broj2      ⎪
      ⎩ broj        ⎭
```

pri čemu su *broj*, *broj1*, *broj2* celi brojevi koji određuju liniju ili linije programa koje treba prikazati.

Listanje se uobičajeno vrši na ekranu, ali se korišćenjem CMD naredbe ono može ostvariti i na nekoj eksternoj jedinici, na primer — na štampaču. Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

DELETE (Basic 7.0)

Koristi se za brisanje dela programa smeštenog u memoriji računara. Opšti oblik komande je:

```
        ⎧ —broj2      ⎫
DELETE  ⎨ broj1—      ⎬
        ⎪ broj1—broj2 ⎪
        ⎩ broj        ⎭
```

pri čemu su *broj*, *broj1* i *broj2* celi brojevi koji određuju liniju ili linije programa koje treba brisati.

Ova komanda se može koristiti isključivo u direktnom načinu rada.

RENUMBER (Basic 7.0)

Koristi se za promenu obeležja naredbi programa ili dela programa smeštenog u memoriji računara. Opšti oblik komande je:

RENUMBER [broj1 [,broj2 [,broj3]]]

pri čemu je:

broj1 — celobrojna vrednost koja označava početnu vrednost obeležja naredbi u programu (ili delu programa) nakon izvršenja komande RENUMBER. Parametar je opcionog karaktera, i ukoliko se izostavi — podrazumeva se vrednost 10,

broj2 — celobrojna vrednost koja određuje uvećanje između dva susedna obeležja naredbi. Parametar može biti izostavljen, a podrazumevana vrednost je 10,

broj3 — celobrojna vrednost koja određuje obeležje neke linije u programu od koje počinje da se izvršava komanda RENUMBER. Parametar je opcionog karaktera, a ukoliko se izostavi — podrazumeva se brojna vrednost koja odgovara obeležju prve naredbe u programu.

Ovu komandu je moguće koristiti isključivo u direktnom načinu rada.

KEY (Basic 7.0)

Služi za definisanje funkcionalnih tastera (F1, F2, ..., F8) ili prikaz funkcija koje su im dodeljene. Opšti oblik komande je:

KEY [broj,sadržaj]

pri čemu je:

broj — celobrojna vrednost iz intervala [1,8] koja definiše jedan od funkcionalnih tastera F1,F2, ...,F8.

sadržaj — podatak, komanda ili niz komanda koje će biti dodeljene tasteru. Pritiskom na neki od funkcionalnih tastera na ekranu se prikazuje sadržaj koji mu je dodeljen. Ovaj parametar može biti maksimalno 241 karakter.

Komanda KEY, u kojoj su izostavljeni parametri *(broj i sadržaj)*, služi za prikaz (listanje) funkcija koje su dodeljene funkcionalnim tasterima.

Napomene:

1) Od 8 raspoloživih tastera, 4 se aktiviraju direktnim pritiskom na odgovarajući taster (F1, F3, F5 i F7) a preostala 4 (F2, F4, F6 i F8) pritiskom na jedan od tastera F1, F3, F5, F7 u kombinaciji sa tasterom SHIFT.

2) Da bi se pritiskom na neki od funkcionalnih tastera izvršila funkcija ili operacija koja mu je dodeljena, neophodno je da zahtevana funkcija ima na kraju ugrađen kôd za RETURN, tj. SHR$(13).

Komanda KEY se može koristiti bilo u direktnom načinu rada bilo na nivou programa.

CONT (Basic 2.0)

Omogućuje nastavljanje privremeno prekinutog programa. Do prekida je moglo doći zato što se pri izvršenju programa naišlo na naredbu STOP ili END, ili je u toku izvršenja programa pritisnut taster RUN/STOP. Opšti oblik naredbe je

CONT

Nakon izvršenja ove komande nastavlja se sa izvršenjem programa od mesta na kojem je izvršen prekid. Ova komanda se može koristiti isključivo u direktnom načinu rada.

NEW (Basic 2.0)

Služi za brisanje programa i korišćenih promenljivih iz memorije računara. Opšti oblik komande je

NEW

Komanda NEW se uglavnom koristi kada treba izbrisati prethodni sadržaj memorije — kako bi se uneo novi program.

Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

RUN (Basic 2.0)

Koristi se za aktiviranje programa smeštenog u memoriji računara. Opšti oblik komande je

RUN [broj]

pri čemu je:

broj — parametar koji može biti izostavljen. Ukoliko se navede, on definiše obeležje programskog reda od koga započinje izvršavanje programa. U protivnom, program se izvršava od prve linije programa.

Napomena: U BASIC-u 7.0 postoji još jedan oblik komande RUN. U ovom slučaju opšti oblik komande je

RUN "naziv" [,Ddisk][,Ujedinica]

pri čemu je:

naziv — parametar koji definiše naziv programa smeštenog na disketi.

Parametri *disk* i *jedinica* imaju isto značenje kao u slučaju naredbe DOPEN.

Ovaj oblik komande RUN omogućuje učitavanje naznačenog programa sa diskete u memoriju računara, a zatim aktiviranje njegovog izvršenja.

Komanda RUN se može izvršavati bilo u direktnom načinu rada, ćelo na nivou programa

Primeri

Primer 1 — Korišćenje naredbi, REM, LET i STOP

```
10 REM
20 REM IZRACUNAVANJE OBIMA I POVRSINE KRUZNICE
30 REM O=2*PI*R   P=PI*R*R
40 INPUT "POLUPRECNIK KRUZNICE JE ",R
50 LET O=2*PI*R
60 LET P=PI*R*R
70 PRINT "OBIM ",O
80 PRINT "POVRSINA ",P
90 STOP
```

Programski redovi 10, 20 i 30 predstavljaju komentar koji nema značaja za izvršenje programa. Linije komentara su veoma korisne u slučaju većih programa jer program čine čitljivijim

Programskim redom 40 prihvata se vrednost koja predstavlja poluprečnik kružnice

Programski redovi 50 i 60 služe za izračunavanje obima i površine pri čemu je službena reč LET mogla biti izostavljena. Programske linije 70 i 80 obezbeđuju prikaz rezultata, a poslednja linija označava fizički kraj programa

Primer 2 — Korišćenje komandi CONT i RUN, kao i naredbe STOP

Izvršimo izmenu u prethodnom programu dopisujući novi programski red

```
75 STOP
```

Ponovnim aktiviranjem programa, nakon unošenja željene vrednosti za promenljivu R, na ekranu se dobija sračunata vrednost obima a u sledećem redu ekrana poruka

```
BREAK IN 75
READY
```

koja je prouzrokovana naredbom STOP u programskom redu 75. U ovom trenutku možemo, na primer, u direktnom načinu rada zahtevati prikaz vrednosti promenljive R naredbom

```
PRINT "POLUPRECNIK ",R
```

a zatim, nakon dobijanja ove vrednosti, komandom CONT obezbediti dalje izvršenje programa, što kao rezultat daje vrednost površine i poruku READY

U ovom trenutku možemo izvršiti još jedan eksperiment. Zahtevamo iz direktnog načina rada prikaz sadržaja promenljivih R, O i P. Dobićemo vrednosti sa kojima smo operisali u poslednjoj realizaciji programa. Aktivirajmo sada komandu:

```
RUN 70
```

kojom se zahteva izvršenje programa smeštenog u memoriji računara, ali počev od programskog reda 70. Na ekranu ćemo dobiti

```
OBIM 0
BREAK IN 75
READY
```

Nastavak programa realizovaćemo komandom CONT, a kao rezultat dobićemo:

```
POVRSINA 0
READY
```

Dobijeni rezultati potvrđuju da se pri aktiviranju programa vrednosti svih promenljivih postavljaju na početno stanje (nule ili blanko, u zavisnosti od tipa promenljive). U konkretnom slučaju to je vrednost 0.

Primer 3 — Korišćenje naredbe WAIT

```
10 PRINT "POCETAK"
20 PRINT "PROGRAM JE U STANJU CEKANJA DA SE
30 PRINT "NA KASETOFONU PRITISNE TASTER ZA AKTIVI-
   RANJE TRAKE"
40 WAIT 1,32,32
50 PRINT "NASTAVAK IZVRSENJA PROGRAMA"
60 STOP
```

Aktiviranjem programa na ekranu se prikazuju poruke određene programskim redovima 10, 20 i 30. Nakon toga, naredbom WIAT u programskoj liniji 40 prekida se dalje izvršenje programa. Aktiviranjem odgovarajućeg tastera na kasetofonu nastavlja se sa izvršenjem programa, o čemu se dobija odgovarajuća poruka (programska linija 50).

Primer 4 — Korišćenje komandi AUTO, RENUBER, LIST, NEW KEY

Napisaćemo jedan elementaran program koji će za dva zadata broja izvršiti četiri osnovne operacije. Pri pisanju programa koristićemo komandu AUTO. Upišimo najpre

```
AUTO 5
```

Ovom komandom aktivirali smo automatsko numerisanje programskih linija. Nakon toga upišimo:

```
100 INPUT "PRVI OPERAND: ",A
```

Odmah nakon pritiska tastera RETURN, u sledećem redu ekrana automatski će se pojaviti obeležje 105. Od nas se očekuje da upišemo željeni sadržaj programskog reda i pritisnemo taster RETURN. Upišimo.

```
INPUT "DRUGI OPERAND: ",B
```

tako da sada programska linija ima izgled:

```
105 INPUT "DRUGI OPERAND: ",B
```

Nakon pritiska RETURN, u sledećem redu ekrana pojaviće se obeležje 110, i tako redom. Pri tome upišimo sledeće naredbe, svaku u posebnom redu:

```
PRINT "ZBIR: ",A+B
PRINT "RAZLIKA: ",A—B
PRINT "PROIZVOD: ",A*B
PRINT "KOLICNIK: ",A/B
STOP
```

Ovim naredbama prethode redom obeležja 110, 115, 120, 125 i 130. Iza toga, u sledećem redu ekrana pojavljuje se obeležje 135. Da bismo prekinuli dalje upisivanje programa, potrebno je samo pritisnuti taster RETURN.

Aktivirajmo prethodno uneti program, a kao operande upišimo 5 i —2. Dobićemo rezultat:

```
ZBIR: 3
RAZLIKA: 7
PROIZVOD: —10
KOILICNIK: —2.5
```

Iskoristimo sada komandu RENUMBER. U tom cilju upišimo

```
RENUMBER 120,10
```

kojom će biti izvršena promena obeležja naredbi čitavog programa. Obeležje prve programske linije biće 120, a svake naredne veće za 10. Da bismo na ekranu prikazali listu programa smeštenog u memoriji računara, upišimo komandu:

```
LIST
```

Na ekranu dobijamo

```
120 INPUT "PRVI OPERAND ",A
130 INPUT "DRUGI OPERAND ",B
140 PRINT "ZBIR ",A+B
150 PRINT "RAZLIKA ",A—B
160 PRINT "PROIZVOD ",A*B
170 PRINT "KOLICNIK ",A/B
180 STOP
```

Ponovnim aktiviranjem programa za iste vrednosti operanada (A=5 B=—2) dobijamo iste izlazne rezultate

Kao što je ranije rečeno, odmah nakon uključivanja računara funkcionalni tasteri F1,F2 ... ,F8 imaju pridružene određene funkcije U BASIC-u računara C-128 postoji komanda KEY kojom je moguće prikazati ove inicijalno dodeljene funkcije U tu svrhu otkucajmo

```
KEY
```

i na ekranu se dobija

```
KEY 1,"GRAPHIC"
KEY 2,"DLOAD"+CHR$(34)
KEY 3,"DIRECTORY"+CHR$(13)
KEY 4,"SCNCLR"+CHR$(13)
KEY 5,"DSAVE"+CHR$(34)
KEY 6,"RUN"+CHR(13)
KEY 7,"LIST"+CHR$(13)
KEY 8,"MONITOR"+CHR$(13)
```

U ovom trenutku nećemo ulaziti detaljno u značenje pojedinih tastera. Zadržaćemo se isključivo na tasterima F6 i F7 Njima su respektivno definisane komande RUN i LIST Prema tome, pomenute dve komande, koje smo već koristili u ovom primeru, možemo aktivirati i korišćenjem funkcionalnih tastera F6 i F7 Primera radi, pritisnimo funkcionalni taster F7 Na ekranu će se prikazati komanda LIST a odmah zatim izvršiće se listanje programa smeštenog u memoriji računara

Iskoristićemo sada drugu mogućnost koju pruža komanda KEY, da bismo promenili značenje određenih funkcionalnih tastera Recimo, želimo da se program automatski poziva u izvršenje pritiska na taster F1 U tu svrhu upišimo komandu

```
KEY 1,"RUN"+CHR$(13)
```

Pritiskom na taster F1 konstatujemo da je zahtevana funkcija realizovana Dalje, ako pritiskom na taster F2 treba brisati program smešten u memoriji računara to se može realizovati sa

```
KEY 2,"NEW"+CHR$(13)
```

Pritiskom tastera F2, tj SHIFT/F1, biće obrisan program smešten u memoriji računara. U ovo se možemo uveriti ako pokušamo da izlistamo program koji je prethodno bio u memoriji računara

Primer 5 — Korišćenje naredbe SLEEP i komande DELETE

```
10 PRINT "POCETAK PROGRAMA"
20 PRINT "KOLIKO SEKUNDI ZAUSTAVITI IZVRSENJE PRO-
   GRAMA?"
30 INPUT S
40 SLEEP S
50 PRINT "NASTAVAK PROGRAMA"
60 PRINT "FUNKCIJE DODELJENE TASTERIMA F1—F8 SU "
70 KEY
80 PRINT  PRINT "KRAJ PROGRAMA"
90 END
```

Aktivirajmo program i kao ulazni parametar upišimo 7 Ovim smo zahtevali da se zaustavi dalje izvršavanje programa u trajanju od 7 sekundi (što je obezbeđeno programskim redom 40) Nakon isteka tog vremena nastavice se dalje izvršenje programa kojim je obezbeđeno prikazivanje funkcija dodeljenih funkcionalnim tasterima

Aktivirajmo ponovo program i kao ulaznu veličinu upišimo 15 Uočava se da je vreme zaustavljanja programa znatno duže

Pretpostavimo da iz programa treba izbrisati određene programske redove, i to 20—50, kao i programski red 80 To možemo realizovati komandama

```
DELETE 20—50
DELETE 80
```

što možemo neposredno proveriti listanjem programa smeštenog u memoriji računara

5.4 RASPOLOŽIVE FUNKCIJE

Na računaru C-128, u okviru BASIC jezika dostupan je veći broj funkcija Pomenute funkcije ne egzistiraju samostalno, već su sastavni deo naredbi Mogu se podeliti na

a) funkcije koje se koriste u matematici,
b) funkcije koje se koriste u radu sa znakovnim podacima,
c) funkcije koje se koriste u radu sa ekranom
d) sistemske funkcije,
e) funkcije koje se koriste u radu sa grafikom,
f) funkcije koje se koriste u radu sa sprajtovima,
g) funkcije koje se koriste u radu sa palicom ili svetlosnim perom

Pre nego što pristupimo razmatranju svake od navedenih grupa funkcija reći ćemo nešto o postojećim sistemskim promenljivim koje

se mogu koristiti na računaru C-128. Pod sistemskom promenljivom podrazumevamo promenljivu, definisanu od strane sistema, koja ima unapred određeno značenje, i to

- ST — promenljiva čija vrednost ukazuje na status ulazno/izlazne operacije,
- TI — promenljiva čija vrednost ukazuje na proteklo vreme internog merača vremena
- TI$ — promenljiva koja služi za postavljanje internog merača vremena ili dobijanje vremena po formatu hh/mm/ss,
- EL — promenljiva čija vrednost ukazuje na obeležje programskog reda u kome je otkrivena greška u toku izvršenja programa,
- ER — promenljiva čija vrednost definiše kôd greške otkrivene pri izvršavanju programa
- ERR$ — promenljiva čija vrednost definiše poruku o grešci koja je otkrivena pri izvršavanju programa,
- DS — promenljiva čija vrednost predstavlja status komandnog kanala jedinice diskete,
- DS$ — promenljiva čija vrednost predstavlja status (kôd statusa i odgovarajuća poruka) komandnog kanala jedinice diskete.

Funkcije koje se koriste u matematici

Ovoj grupi pripadaju funkcije SQR, EXP, LOG, ABS, SGN, INT, SIN COS TAN ATN, DEC NEX$, RND, XOR i FN.
Treba znati da

— za bilo koju od funkcija SQR EXP, LOG SIN, COS TAN, ATN i FN u najvećem broju slučajeva dobijena vrednost je približne tačnosti,

— izračunavanje funkcije predstavlja operaciju najvećeg prioriteta,

— u slučaju trigonometrijskih funkcija ugao se zadaje u radijanima.

SQR (Basic 2.0)

Koristi se za izračunavanje kvadratnog korena neke brojne vrednosti. Opšti oblik funkcije je

SQR (broj)

pri čemu je *broj* brojni izraz kako je to definisano u odeljku 5.1, uz napomenu da vrednost ovog parametra ne sme biti manja od nule.

EXP (Basic 2.0)

Služi za određivanje eksponencijalne vrednosti brojnog izraza. Opšti oblik funkcije je

EXP (broj)

pri čemu je parametar *broj* brojni izraz definisan u odeljku 5.1. Ovaj parametar ima vrednost reda veličine [—88.88].

Napomena. Reč je o eksponencijalnoj funkciji čiji je matematički zapis e^{broj}, pri čemu je e=2.71828183.

LOG (Basic 2.0)

Koristi se za određivanje logaritma nekog brojnog izraza. Pri tome se misli na matematičku funkciju *ln(x)*, čija je osnova e=2.718... Opšti oblik funkcije je

LOG (broj)

pri čemu se pod pojmom *broj* podrazumeva brojni izraz kako je to definisano u odeljku 5.1, uz napomenu da vrednost pomenutog parametra mora biti veća od nule.

ABS (Basic 2.0)

Služi za određivanje apsolutne vrednosti brojnog izraza. Opšti oblik funkcije je

ABS (broj)

pri čemu parametar *broj* predstavlja brojni izraz kako je to definisano u odeljku 5.1.

Napomena. Po definiciji je

$$|x| = \begin{cases} x, & \text{za } x>0 \\ 0, & \text{za } x=0 \\ -x, & \text{za } x<0 \end{cases}$$

SGN (Basic 2.0)

Koristi se za određivanje znaka brojnog izraza. Opšti oblik funkcije je

SGN (broj)

pri čemu parametar *broj* predstavlja brojni izraz kako je to definisano u odeljku 5.1.

Napomena. Po definiciji je

$$\text{sgn}(x) = \begin{cases} 1, & \text{za } x>0 \\ 0, & \text{za } x=0 \\ -1, & \text{za } x<0 \end{cases}$$

INT (Basic 2.0)

Služi za određivanje celobrojne vrednosti nekog brojnog izraza. Opšti oblik funkcije je:

INT (broj)

pri čemu parametar *broj* predstavlja brojni izraz kako je to definisano u odeljku 5.1

SIN (Basic 2.0)

Koristi se za određivanje vrednosti trigonometrijske funkcije *sin(x)* za vrednost naznačenog brojnog izraza. Opšti oblik funkcije je:

SIN (broj)

pri čemu parametar *broj* predstavlja vrednost ugla izraženog u radijanima.

Napomena. Za izračunavanje vrednosti trigonometrijskih funkcija posebno je značajna konstanta π, koja ima vrednost 3,14159265. Ova konstanta postoji definisana na računaru C-128 i može se dobiti istovremenim pritiskom tastera SHIFT i ↑ (koji je smešten s leve strane tastera RESTORE).

COS (Basic 2.0)

Služi za određivanje vrednosti trigonometrijske funkcije *cos(x)* za vrednost naznačenog brojnog izraza. Opšti oblik funkcije je

COS (broj)

pri čemu parametar *broj* predstavlja vrednost ugla izraženu u radijanima.

TAN (Basic 2.0)

Koristi se za određivanje vrednosti trigonometrijske funkcije *tg(x)* za vrednost naznačenog brojnog izraza. Opšti oblik funkcije je

TAN (broj)

pri čemu parametar *broj* predstavlja vrednost ugla izraženog u radijanima.

Napomena:

a) Pri korišćenju funkcije *tg(x)* treba voditi računa o oblasti definisanosti pomenute funkcije.

b) Funkcija *ctg(x)* se može realizovati korišćenjem relacije ctg(x)= =1/tg(x)

ATN (Basic 2.0)

Služi za određivanje vrednosti inverzne trigonometrijske funkcije *arctg(x)* za vrednost naznačenog brojnog izraza. Opšti oblik funkcije je

ATN (broj)

pri čemu parametar *broj* predstavlja brojni izraz kako je to definisano u odeljku 5.1

DEC (Basic 7.0)

Koristi se za prevođenje (konverziju) heksadecimalne u decimalnu brojnu vrednost. Opšti oblik funkcije je

DEC (promenljiva)

pri čemu parametar *promenljiva* predstavlja znakovnu promenljivu kojom je zadata određena vrednost u heksadecimalnom zapisu

Napomena

a) Pod heksadecimalnim brojnim sistemom podrazumevamo brojni sistem sa osnovom 16, čije su "cifre" 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

b) Vrednost parametra *promenljiva* mora biti iz opsega [0,FFFF].

HEX$ (Basic 7.0)

Služi za prevođenje decimalne u heksadecimalnu brojnu vrednost. Opšti oblik funkcije je

HEX$ (broj)

pri čemu parametar *broj* mora imati vrednost iz intervala [0,65535] koju treba pretvoriti u heksadecimalni zapis.

RND (Basic 2.0)

Koristi se za generisanje slučajnog broja iz intervala [0,1]. Opšti oblik funkcije je

RND (broj)

pri čemu parametar *broj* predstavlja brojni izraz kako je to definisano u odeljku 5.1

Funkcijom RND generiše se pseudo-slučajan broj na osnovu znaka i vrednosti argumenata. Pri tome znak argumenta definiše jedan od 3 moguća načina izbora početne (tzv. *seed*) vrednosti, koja služi kao osnov za sračunavanje slučajnog broja. Ukoliko je argument

5*

a) negativan — tada se polazi od određene *seed* vrednosti u zavisnosti od vrednosti argumenta,

b) nula — tada se *seed* vrednost uzima na osnovu vrednosti sistemskog sata,

c) pozitivan — tada vrednost argumenta nema značaja na izbor *seed* vrednosti (zadržava se postojeća *seed* vrednost)

XOR (Basic 7.0)

Funkcija čiji je rezultat brojna vrednost dobijena operacijom ekskluzivno OR nad zadatim brojnim argumentima. Opšti oblik funkcije je

XOR(broj1,broj2)

pri čemu su *broj1* i *broj2* neoznačene celobrojne vrednosti iz intervala [0,65535].

FN (Basic 2.0)

Koristi se za dobijanje brojne vrednosti koja predstavlja vrednost funkcije koja je definisana u programu (naredbom DEF FN, o kojoj će kasnije biti reči). Opšti oblik funkcije je:

FN funkcija(broj)

pri čemu je *funkcija* parametar koji definiše naziv novodefinisane funkcije, a argument *broj* je brojna vrednost za koju treba izračunati vrednost funkcije.

Funkcije koje se koriste u radu sa znakovnim podacima

Ovoj grupi pripadaju funkcije ASC, CHR$, LEFT$, RIGHT$, MID$, STR$, VAL, LEN i INSTR.

ASC (Basic 2.0)

Koristi se za prevođenje prvog karaktera zadatog niza/promenljive u odgovarajući ASCII kôd. Opšti oblik funkcije je

ASC (izraz)

pri čemu parametar *izraz* predstavlja znakovni izraz kako je to definisano u odeljku 5.1. Rezultat funkcije predstavlja brojnu vrednost iz opsega [0,255].

CHR$ (Basic 2.0)

Služi za prevođenje brojnog u znakovni podatak. Opšti oblik funkcije je

CHR$(broj)

pri čemu parametar *broj* predstavlja brojni izraz kako je to definisano u odeljku 5.1

LEFT$ (Basic 2.0)

Koristi se za dobijanje podniza sastavljenog od određenog niza karaktera na početku zadatog podatka/promenljive. Opšti oblik funkcije je:

LEFT$(niz,broj)

pri čemu parametar *niz* predstavlja znakovni podatak/promenljivu, a *broj* celobrojnu vrednost iz opsega [0,255]

RIGHT$ (Basic 2.0)

Koristi se za dobijanje podniza koji čini određeni broj karaktera sa desnog kraja zadatog podatka/promenljive. Opšti oblik funkcije je

RIGHT$(izraz,broj)

pri čemu parametri *izraz* i *broj* imaju isto značenje kao u slučaju funkcije **LEFT$**

MID$ (Basic 2.0)

Služi za dobijanje podniza iz zadatog niza. Opšti oblik funkcije je:

MID$ (izraz,broj1[,broj2])

pri čemu je *izraz* znakovni podatak ili promenljiva, *broj1* predstavlja brojni izraz čija vrednost definiše početnu poziciju, a *broj2* dužinu traženog podniza. Argumenti *broj1* i *broj2* moraju biti iz opsega [0,255]

Napomena. U C-128 môdu funkcija **MID$** ima još jednu mogućnost. Naime, može koristiti supstituciju jednog podniza u okviru datog niza određenim sadržajem.

STR$ (Basic 2.0)

Koristi se za prevođenje brojnog u znakovni podatak. Opšti oblik funkcija je:

STR$(broj)

pri čemu je *broj* brojni izraz kako je to definisano u odeljku 5.1

VAL (Basic 2.0)

Koristi se za pretvaranje znakovnog u brojni podatak. Opšti oblik funkcije je:

VAL(izraz)

pri čemu pod pojmom *izraz* podrazumevamo znakovni izraz kako je to definisano u odeljku 5.1. Ova funkcija je inverzna funkciji STR$.

LEN (Basic 2.0)

Služi za određivanje dužina jednog znakovnog podatka/izraza, tj. broja znakova koji ga sačinjavaju. Opšti oblik funkcije je

LEN(izraz)

pri čemu pod pojmom *izraz* podrazumevamo znakovni podatak/izraz kako je to definisano u odeljku 5.1.

INSTR (Basic 7.0)

Koristi se za određivanje početne pozicije jednog niza u okviru drugog. Opšti oblik funkcije je

INSTR (niz1,niz2[,broj])

pri čemu je

niz1 — parametar koji definiše znakovni podatak u okviru koga se pretražuje znakovni podatak određen parametrom *niz2*

niz2 — parametar koji definiše znakovni podatak koji se pretražuje u okviru znakovnog podatka određenog parametrom *niz1*,

broj — parametar koji definiše poziciju u okviru znakovnog podatka *niz1*, od koje počinje pretraživanje znakovnog podatka *niz2*. Ovaj parametar je opcionog karaktera i ukoliko se izostavi — podrazumeva se vrednost 1. Može biti u opsegu 1—225

Funkcije koje se koriste u radu sa ekranom

Ovoj grupi pripadaju funkcije POS, RWINDOW, SPC i TAB

POS (Basic 2.0)

Koristi se za određivanje tekuće pozicije kursora, tj. kolone u kojoj se nalazi kursor. Opšti oblik funkcije je

POS(argument)

pri čemu *argument* može biti bilo koji brojni ili znakovni izraz.

Napomena: U BASIC-u 2.0 pod linijom ekrana se podrazumevaju dve fizičke linije. S obzirom na to da je ekran 40-to kolonski, funkcija POS kao rezultat daje brojnu vrednost iz opsega 0—79.

U BASIC-u 7.0, vrednost koju daje funkcija POS može biti iz opsega 0—39 (ukoliko se koristi 40-to kolonski ekran), odnosno iz opsega 0—79 (ukoliko se koristi 80-to kolonski ekran).

RWINDOW (Basic 7.0)

Služi za određivanje osnovnih parametara tekućeg prozora ekrana (pod pojmom prozora podrazumevamo deo ekrana koji je definisan za radno područje). Opšti oblik funkcije je

RWINDOW(broj)

pri čemu je *broj* celobrojna vrednost koja može biti 0,1 ili 2, i to

0 — ukazuje da funkcija kao rezultat daje broj kolona u tekućem prozoru,

1 — ukazuje da funkcija kao rezultat daje broj linija u tekućem prozoru,

2 — ukazuje da funkcija kao rezultat daje dužinu linija ekrana, koja može biti 40 ili 80

SPC (Basic 2.0)

Koristi se za pomeranje kursora za određeni broj pozicija udesno — u odnosu na njegovu tekuću poziciju. Opšti oblik funkcije je

SPC(broj)

pri čemu parametar *broj* predstavlja vrednost koja može biti iz opsega 0—255

Ova funkcija se može koristiti isključivo u kombinaciji sa naredbama PRINT i PRINT# i služi pri formatizovanju izlaznih podataka bilo da je izlazna jedinica ekran bilo neka druga eksterna jedinica

TAB (Basic 2.0)

Služi za pomeranje kursora udesno za naznačeni broj pozicija u tekućoj liniji, računajući od prve pozicije. Opšti oblik funkcije je

TAB(broj)

pri čemu parametar *broj* ima vrednost iz opsega 0—255 Ova funkcija se može koristiti isključivo u okviru naredbe PRINT

Sistemske funkcije

Ovoj grupi pripadaju funkcije ERR$, FRE, PEEK, POINTER i USR

ERR$ (Basic 7.0)

Ova funkcija daje kao rezultat odgovarajuću poruku o grešci — u zavisnosti od argumenta. Opšti oblik funkcije je

ERR$(broj)

pri čemu je *broj* celobrojna vrednost iz opsega [1,41] koja predstavlja kôd greške

Napomena. Najčešće se kao argument funkcije koristi sistemska promenljiva *ER*.

FRE (Basic 2.0)

Koristi se za dobijanje informacije o raspoloživoj RAM memoriji namenjenoj programu i promenljivim u njemu. Opšti oblik funkcije je

FRE(argument)

pri čemu je *argument* — parametar koji ima sledeće osobine

a) U BASIC-u 2.0 pomenuti parametar može biti bilo koji brojni ili znakovni podatak i on nema uticaja na vrednost funkcije FRE. Kao rezultat funkcije FRE dobija se brojna vrednost koja definiše raspoloživu RAM memoriju koja se može koristiti bilo za program, bilo za promenljive. Pri tome, dobijena vrednost može biti i negativna (ukoliko je broj slobodnih bajtova veći od 32767). Da bi se u ovom slučaju dobila tačna vrednost raspoložive memorije, koristi se izraz

FRE(0) — 65535*(FRE(0)<0)

b) U BASIC-u 7.0 pomenuti parametar može imati vrednost 0 ili 1, tako da FRE(0) određuje veličinu raspoložive memorije namenjene programu, a FRE(1) veličinu raspoložive memorije namenjenu promenljivim.

PEEK (Basic 2.0)

Koristi se za dobijanje sadržaja sa određene memorijske lokacije. Opšti oblik funkcije je

PEEK(broj)

pri čemu parametar *broj* predstavlja brojni izraz čija vrednost definiše memorijsku lokaciju i može biti iz opsega [0,65535]. Rezultat funkcije je celobrojna vrednost iz opsega [0,255]

Napomena. Funkcija PEEK radi u okviru poslednje odabrane banke (što se realizuje komandom BANK). Ova osobina važi isključivo u C-128 režimu rada, budući da u C-64 načinu rada ne postoji mogućnost izbora banke, tj. određene memorijske konfiguracije.

POINTER (Basic 7.0)

Služi za određivanje memorijske lokacije na kojoj je smeštena određena promenljiva. Opšti oblik funkcije je

POINTER(promenljiva)

pri čemu parametar **promenljiva** definiše naziv brojne/znakovne promenljive za koju se određuje memorijska lokacija.

USR (Basic 2.0)

Koristi se za pozivanje mašinskog programa iz BASIC programa, pri čemu početnu adresu mašinskog programa treba smestiti na memorijskim lokacijama 785 i 786 (u C 64 režimu rada), odnosno 4633 i 4634 (u C-128 režimu rada), korišćenjem naredbe POKE. Opšti oblik funkcije je

USR(broj)

pri čemu *broj* predstavlja parametar koji se prenosi u mašinski program. Nakon izvršenja mašinskog programa dobijeni rezultat je dostupan na nivou BASIC programa kao vrednost funkcije USR.

Funkcije koje se koriste u radu sa grafikom

Ovoj grupi pripadaju funkcije RCLR, RDOT i RGR. Pomenute funkcije biće detaljno obrađene u okviru poglavlja koje se odnosi na grafiku, a za sada ćemo navesti samo njihovu osnovnu namenu.

RCLR — koristi se za određivanje kôda boje određenog dela ekrana,

RDOT — koristi se za određivanje tekuće pozicije grafičkog kursora, ili boje kursora,

RGR — koristi se za određivanje tekućeg grafičkog môda ekrana.

Funkcije koje se koriste u radu sa sprajtovima

Ovoj grupi pripadaju funkcije BUMP, RSPCOLOR, RSPRITE i RSPPOS. One će biti detaljno obrađene u okviru poglavlja koje se odnosi na sprajtove, a za sada navodimo samo njihovu osnovnu namenu.

BUMP — koristi se za dobijanje informacija o sudaru sprajtova ili sprajtova i pozadine,

RSPCOLOR — koristi se za dobijanje sadržaja kolor registara koje koriste sprajtovi,

RSPRITE — koristi se za dobijanje osnovnih informacija o sprajtovima,

RSPPOS — koristi se za dobijanje informacija o poziciji sprajta i njegovoj brzini.

Funkcije koje se koriste u radu sa palicom i svetlosnim perom

Ovoj grupi pripadaju funkcije JOY, POT i PEN.

JOY (Basic 7.0)

Služi za određivanje pozicije palice za igru i stanja tastera za paljbu. Opšti oblik funkcije je

JOY(broj)

pri čemu parametar *broj* može imati vrednost 1 ili 2 koja ukazuje na odgovarajuću palicu za igru (port 1 ili port2).

Vrednost funkcije JOY, shodno položaju palice, može biti iz opsega [0,8] prema šemi



Napomena: Ako je istovremeno pritisnut taster za paljbu tada se svaka vrednost u prethodnoj šemi uvećava za 128

POT (Basic 7.0)

Koristi se za dobijanje vrednosti potenciometra (tzv "paddle") koji je priključen na mestu predviđenom za palicu za igru Opšti oblik funkcije je

POT(broj)

pri čemu parametar *broj* može biti iz opsega [1,4] i on definiše jedan od četiri moguća položaja potenciometra

Vrednost funkcije može biti iz opsega [0,255] Ukoliko je pri tom pritisnut i taster za paljbu, tada se odgovarajuća vrednost uvećava za 256

PEN (Basic 7,0)

Služi za određivanje tekućih koordinata svetlosne olovke Opšti oblik je

PEN(broj)

pri čemu parametar *broj* može biti iz opsega [0 4] i ima sledeće značenje

0 — vrednost funkcije predstavlja X-koordinatu tekuće pozicije svetlosne olovke (40-to kolonski ekran),

1 — vrednost funkcije predstavlja Y-koordinatu tekuće pozicije svetlosne olovke (40-to kolonski ekran)

2 — vrednost funkcije predstavlja X-koordinatu tekuće pozicije svetlosne olovke (80-to kolonski ekran)

3 — vrednost funkcije predstavlja Y-koordinatu tekuće pozicije svetlosne olovke (80-to kolonski ekran),

4 — vrednost funkcije je aktivirana vrednost svetlosne olovke

Primeri

Primer 1 — Izračunavanje kvadratnog korena iz zadatog broja

```
10 INPUT "BROJNA VREDNOST " A
20 PRINT "ZA A= " A " KVADRATNI KOREN JE " SQR(A)
```

Primer 2 — Izračunavanje vrednosti $log_b(a)$, pri čemu su a i b zadate vrednosti

```
10 INPUT "OSNOVA LOGARITMA ",B
20 INPUT "VREDNOST ZA KOJU SE ODREDJUJE LOGARI-
   TAM ",A
30 PRINT "REZULTAT " LOG(A)/LOG(B)
```

Za ulazne veličine A=8 i B=2 dobija se rezultat 3

Primer 3 — Izračunavanje vrednosti izraza e^a za zadatu vrednost a

```
10 INPUT "PODATAK A=",A
20 PRINT "REZULTAT ",EXP(A)
```

Primer 4 — Izračunavanje apsolutne vrednosti izraza $a-b$, pri čemu su a i b zadate brojne vrednosti

```
10 INPUT "PODACI A, B SU " A,B
20 PRINT "APSOLUTNA VREDNOST RAZLIKE JE ",ABS(A—B)
```

Napomena. Nakon aktiviranja programa i dobijanja poruke "PODACI A, B SU" potrebno je upisati željenu vrednost podatka A, a zatim simbol *zarez* i vrednost podatka B. Na kraju unosa treba pritisnuti taster RETURN

Primer 5 — Izračunavanje znaka proizvoda $a \cdot b$, pri čemu su a i b zadate vrednosti

```
10 INPUT "PODATAK A=" A
20 INPUT "PODATAK B= " B
30 PRINT "ZNAK PROIZVODA JE " SGN(A*B)
```

Primer 6 — Korišćenje funkcije INT

```
10 INPUT "PODATAK A= " A
20 INPUT "PODATAK B= " B
30 PRINT "A= " A " INT(A)= " INT(A)
40 PRINT "B= " B " INT(B)= " INT(B)
```

Kao ulazne veličine unesimo A=3.19 i B=—8.7. Kao rezultat dobićemo INT(A)=3 INT(B)=—9

Napomena. Pod celobrojnom vrednošću brojnog izraza podrazumeva se najveći ceo broj koji nije veći od vrednosti zadatog brojnog izraza.

Primer 7 — Izračunavanje osnovnih trigonometrijskih funkcija za zadatu vrednost ugla

```
10 INPUT "UGAO, IZRAZEN U STEPENIMA ",X
20 PRINT "SIN(X)=",SIN(PI*X/180)
30 PRINT "COS(X)=",COS(PI*X/180)
40 PRINT "TG(X)=",TAN(PI*X/180)
50 PRINT "CTG(X)=",1/(TAN(PI*X/180)
```

Primer 8 — Korišćenje funkcije ATN

```
10 INPUT "VREDNOST TANGENSA JE ",A
20 PRINT "UGAO U RADIJANIMA ",ATN(A)
```

Primer 9 — Korišćenje funkcija DEC i HEX$

```
10 INPUT "HEKSADECIMALNI (H→D) ",H$
20 INPUT "DECIMALNI (D→H) " D
30 PRINT "H$=",H$, D=",DEC(H$)
40 PRINT "D=",D, H$=",HEX$(D)
```

Primer 10 — Korišćenje funkcije RND za dobijanje celobrojne vrednosti iz opsega [1 6] na slučajan način

```
10 PRINT INT(6*RND(0)+1)
```

Primer 11 — Korišćenje funkcije XOR

```
10 INPUT "PRVI OPERAND";A
20 INPUT "DRUGI OPERAND",B
30 PRINT A,"XOR",B,"=",XOR(A,B)
```

Za ulazne veličine A=40 i B=9, s obzirom na to da je:

$A=(40)_{10}=(00101000)_2$
$B=(9)_{10}==(00001001)_2$

dobija se

$$
\begin{array}{r}
00101000 \\
+\;00001001 \\
\hline
(00100001)_2=(33)_{10}
\end{array}
$$

Primer 12 — Korišćenje funkcija ASC i STR$

```
10 INPUT "ZNAKOVNI PODATAK ",A$
20 INPUT "BROJNI PODATAK ",B
30 PRINT "A$=",A$,"ASC(A$)=",ASC(A$)
40 PRINT "B=",B,"STR$(B)=",STR$(B)
```

Aktiviramo program i kao ulazne veličine upišemo respektivno RACUNAR i 86. Kao rezultat dobićemo

```
A$=RACUNAR      ASC(A$)=82
B=86            STR$(B)=V
```

pri čemu 82 predstavlja ASCII kôd prvog simbola u okviru znakovnog podatka RACUNAR (karaktera R), a V predstavlja karakter čija je ASCII vrednost 86

Primer 13 — Korišćenje funkcija LEFT$, RIGHT$, MID$ i LEN

```
10 A$="EVROPA JUGOSLAVIJA BEOGRAD"
20 PRINT "NA POCETKU:",LEFT$(A$,6)
30 PRINT "U SREDINI:",MID$(A$,8,11)
40 PRINT "NA KRAJU:",RIGHT$(A$,7)
50 PRINT "UKUPNA DUZINA:",LEN(A$)
```

Primer 14 — Korišćenje još jedne mogućnosti funkcije MID$ (raspoložive u C-128 modu, tj. BASIC-u 7.0)

```
10 A$="SVILA": PRINT "A$=",A$
20 MID$(A$,2,3)="ENT"
30 PRINT "A$=",A$
```

U ovom programu se niz VIL, koji se nalazi u okviru niza SVILA, zamenjuje nizom ENT.

Primer 15 — Korišćenje funkcije STR$

```
10 A=—271.48  B=392.59
20 PRINT STR$(A+B)
```

Primer 16 — Korišćenje funkcije VAL

```
10 A$="C128"  B$="1234E56"
20 C$="324.79"  D$="18*3"
30 PRINT "VAL(A$)=",VAL(A$)
40 PRINT "VAL(B$)=",VAL(B$)
50 PRINT "VAL(C$)=",VAL(C$)
60 PRINT "VAL(D$)=",VAL(D$)
```

Primer 17 — Korišćenje funkcije INSTR

```
10 A$="SMEDEREVSKA PALANKA"
20 B$="KA"
30 PRINT INSTR(A$,B$)
40 PRINT INSTR(A$,B$,13)
```

U prvom slučaju pretražuje se sadržaj KA u okviru sadržaja SMEDEREVSKA PALANKA — počev od prve pozicije (programska linija 30). U drugom slučaju (programska linija 40) pretraživanje počinje od pozicije 13. Pri tom se dobijaju redom rezultati 10 i 18.

Primer 18 — Korišćenje funkcije POS

```
10 PRINT "MAKSIMIR" POS(0)
```

Kao rezultat dobija se vrednost 8, što predstavlja tekuću poziciju kolone u kojoj se nalazi kursor.

Napomena: Prva kolona je numerisana sa 0.

Primer 19 — Korišćenje funkcija SPC i TAB

```
10 A$="NAZIV": B$="ADRESA"
20 PRINT A$ SPC(10)B$
30 PRINT A$ TAB(10) B$
```

Primer 20 — Korišćenje funkcije RWINDOW

```
10 WINDOW 1,1,10,20
20 PRINT "BR.LINIJE TEKUCEG PROZORA ",RWINDOW(0)
30 PRINT "BR.KOLONE TEKUCEG PROZORA ",RWINDOW(1)
40 PRINT "FORMAT EKRANA ",RWINDOW(2)
```

Pomenimo da WINDOW predstavlja naredbu za definiciju dela ekrana koji se koristi kao radno područje (tzv. "prozor").

Aktiviranjem prethodnog programa dobijamo:

```
BR.LINIJE TEKUCEG PROZORA  20
BR.KOLONA TEKUCEG PROZORA  10
FORMAT EKRANA  40 (ili 80)
```

Primer 21 — Korišćenje funkcije ERR$

```
10 INPUT "KOD GRESKE (1—41) ";K
20 PRINT "KOD GRESKE ",K
30 PRINT "PORUKA O GRESCI ",ERR$(K)
```

Primer 22 — Korišćenje funkcije FRE

```
10 PRINT "SLOBODNA MEMORIJA NAMENJENA "
20 PRINT " — BASIC PROGRAMU",FRE(0)
30 PRINT " — PROMENLJIVIM ",FRE(1)
```

Primer 23 — Korišćenje funkcije PEEK

```
10 PRINT PEEK(65535)
```

Kao rezultat dobija se sadržaj memorijske lokacije 65535 (u tekućoj banci)

Primer 24 — Korišćenje funkcija POINTER

```
10 A$="PODATAK" B=537
20 PRINT "POINTER ZA A$ ",POINTER(A$)
30 PRINT "POINTER ZA B ",POINTER(B)
```

O praktičnim mogućnostima ove funkcije biće kasnije više reči

Primer 25 — Korišćenje funkcije USR

USR funkcija omogućuje povezivanje mašinskog i BASIC programa. Pri tome, pre pozivanja pomenute funkcije u memoriju računara treba smestiti odgovarajući mašinski program. Nakon toga je potrebno na određene memorijske lokacije (4633—4634 ili 785—786) upisati početnu adresu mašinskog programa. U tu svrhu koristi se naredba POKE (o kojoj će kasnije biti reči). BASIC program ima izgled, na primer:

```
200 B=USR(A)
```

Kada se pri izvršavanju ovog programa naiđe na programsku liniju 200, tada se kontrola prenosi na mašinski program. On vrši obradu nad argumentom A i rezultujući vrednost vraća BASIC programu preko promenljive B. Nakon toga nastavlja se sa izvršavanjem BASIC programa počev od naredbe koja dolazi iza programske linije 200

Primer 26 — Ilustruje korišćenje funkcije FN

```
10 DEF FNF(X)=5*X*X — 2*X + 8
20 INPUT "X=",X
30 PRINT "ARGUMENT X=",X
40 PRINT "FUNKCIJA P(X)=",FNP(X)
```

Pomenimo da naredba DEF FNF(X) služi za definisanje određene funkcije koju želimo da koristimo u programu

5.5. NAREDBE USLOVNOG/BEZUSLOVNOG PRELASKA

Koriste se za kontrolu redosleda izvršavanja naredbi u programu. Ovoj grupi pripadaju naredbe GOTO, ON i IF.

GOTO (Basic 2.0)

Omogućuje bezuslovni prelazak na naredbu sa naznačenim obeležjem, tj. prenosi izvršenje programa na određenu programsku liniju. Opšti oblik naredbe je

GOTO broj

pri čemu je

broj — celobrojna vrednost koja predstavlja obeležje nekog programskog reda,

Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

ON (Basic 2.0)

Koristi se za "grananje" u okviru programa u zavisnosti od vrednosti određenog brojnog izraza. Opšti oblik naredbe je

ON broj $\left\{\begin{matrix} \text{GOTO} \\ \text{GOSUB} \end{matrix}\right\}$ $n_1, n_2, \ldots, n_k$

pri čemu je

broj — brojni izraz kako je to definisano u odeljku 5.1,

$n_1, \ldots, n_k$ — obeležje nekih od programskih redova

Naredba se realizuje na sledeći način. Najpre se određuje celobrojna vrednost izraza definisanog parametrom *broj*. Ukoliko je dobijena vrednost i ($i=1,2, \ldots, k$), tada se vrši skok na programski red sa obeležjem n_i. U protivnom se naredba ignoriše i prelazi se na izvršenje sledeće naredbe u programu.

Ključna reč GOTO se koristi kada je potrebno ostvariti bezuslovni skok, tj. nakon skoka na određeni programski red (koji je realizovan korišćenjem naredbe ON) dalje izvršenje programa se nastavlja počev od tog programskog reda.

Ključna reč GOSUB se koristi ukoliko je potrebno izvršiti određeni broj programskih redova (počev od naznačenog programskog reda, pa sve dok se ne naiđe na naredbu RETURN), a zatim se vratiti na naredbu koja sledi iza naredbe ON i od nje nastaviti dalje izvršenje programa. Ova naredba se može koristiti bilo u direktnom, bilo u odloženom načinu rada.

IF (Basic 2.0 i Basic 7.0)

Omogućuje "grananje" u programu u zavisnosti od naznačenog uslova. Ova naredba ima više oblika, pri čemu u Basic 7.0 postoje oblici koji nisu dostupni u Basic-u 2.0. Stoga ćemo razmatrati dva slučaja.

a) *Basic 2.0* Moguća su dva oblika naredbe, i to:

$$\text{IF uslov THEN} \begin{Bmatrix} \text{broj} \\ \text{naredbe} \end{Bmatrix}$$

IF uslov GOTO broj

pri čemu je

uslov — relacija u kojoj učestvuju dva brojna/znakovna izraza, između kojih stoji neki od simbola upoređivanja (=, <, <=, >, >=, <>). Ovo su tzv. "prosti uslovi". Uslov u IF naredbi može biti i složen. "Složeni uslovi" nastaju kombinacijom prostih uslova, pri čemu se koriste službene reči AND, OR i NOT.

broj — celobrojna vrednost koja predstavlja obeležje nekog programskog reda.

naredbe — jedna ili više naredbi BASIC jezika.

Naredba IF se izvršava na sledeći način:

— Ukoliko se pri izvršavanju programa naiđe na naredbu IF, a pri tom nije ispunjen *uslov*, tada se vrši prelaz na naredbu koja sledi iza IF naredbe.

— Ukoliko se pri izvršavanju programa naiđe na naredbu IF, a pri tom je ispunjen *uslov*, tada se

a) prelazi na programsku liniju sa obeležjem *broj*, ukoliko je iza službene reči THEN (ili GOTO) specificiran ovaj paramatar,

b) izvršavaju naredbe naznačene parametrom *naredbe*, naznačenim iza službene reči THEN. Ukoliko poslednja od njih nije STOP, END ili GOTO, tada se nakon izvršenja IF naredbe prelazi na naredbu koja je sledi.

b) *Basic 7.0.* Pored pomenuta dva moguća su još dva oblika:

$$\text{IF uslov THEN} \begin{Bmatrix} \text{broj} \\ \text{naredbe} \end{Bmatrix} \quad \left[\text{:ELSE} \begin{Bmatrix} \text{broj} \\ \text{naredbe} \end{Bmatrix}\right]$$

```
IF uslov THEN BEGIN
.
.
naredbe
.

BEND [:ELSE BEGIN


naredbe


BEND]
```

Parametri *uslov, broj, naredbe* imaju isto značenje kao u slučaju odgovarajućih parametara IF naredba Basic-a 2.0

U prvom slučaju IF naredba se izvršava na sledeći način. Najpre se ispituje istinitost relacije definisane parametrom *uslov*. Definicija daljeg postupka navedena je iza reči THEN ili iza reči ELSE — u zavisnosti od toga da li je naznačeni *uslov* zadovoljen ili ne. Pri tom postupak može biti određen skupom naredbi koje treba izvršiti ili skokom na određenu programsku liniju.

Napomena. Kompletna IF naredba može biti maksimalno 160 karaktera.

U drugom slučaju IF naredba se izvršava na analogan način kao u prethodnom slučaju. Prednost ovog oblika se ogleda u tome što dozvoljava formiranje struktura na taj način što se između ključnih reči BEGIN i BEND može naći proizvoljan broj programskih linija. Ovde ne postoji ograničenje na dužinu od 160 karaktera.

Naredba IF, bez obzira na oblik o kome je reč, može se koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Primeri

Primer 1 — Program za zadatu vrednost x izračunava vrednost funkcije f(x) po formuli

$$f(x)=\begin{cases} x^2-3x+2, & \text{za } x\geq 0 \\ x+2, & \text{za } x<0 \end{cases}$$

```
10 INPUT "ARGUMENT X=",X
20 IF X<0 THEN Y=X+2: GOTO 40
30 Y=X ↑ 2—3*X+2
40 PRINT "X=";X: PRINT "Y=",Y
50 STOP
```

Primer 2 — Prethodni zadatak rešen korišćenjem IF naredbe iz Basic-a 7.0

```
10 INPUT "ARGUMENT X=",X
20 IF X<0 THEN Y=X+2:ELSE Y=X ↑ 2—3*X+2
30 PRINT "X=",X: PRINT "Y=",Y
40 STOP
```

Primer 3 — Program za dva zadata numerička podatka izvršava naznačenu operaciju, i to:

1 — sabiranje,
2 — oduzimanje,
3 — množenje,
4 — deljenje

```
10 INPUT "PRVI OPERAND X=",X
20 INPUT "DRUGI OPERAND Y=",Y
30 INPUT "OPERACIJA (1,2,3,4) ",OP
40 ON OP GOTO 60,70,80,90
50 GOTO 30
60 PRINT "X+Y=",X+Y  STOP
70 PRINT "X—Y=",X—Y  STOP
80 PRINT "X*Y=",X*Y  STOP
90 PRINT "X/Y=",X/Y  STOP
```

Primer 4 — Pogađanje zamišljenog broja

```
10 A=INT(20*RND(0))+1
20 PRINT "POGODITE ZAMISLENI BROJ (1—20)"
30 INPUT "IZBOR ",I
40 IF I<>A THEN BEGIN
50   IF I<A THEN PRINT "ZAMISLENI BROJ JE VECI"
60   IF I>A THEN PRINT "ZAMISLENI BROJ JE MANJI"
70 BEND  GOTO 30 ELSE PRINT "POGODAK" END
```

Primer 5 — Ilustrovanje još jedne mogućnosti naredbe IF

```
10 INPUT "BROJ B=",B
20 IF B>0 AND B<100 THEN BEGIN
30   PRINT "BROJ ",B
40   PRINT "KVADRAT ",B*B
50   PRINT "KUB ",B*B*B
60 BEND ELSE PRINT "IZVAN OPSEGA"  GOTO 10
70 PRINT "KRAJ PROGRAMA"
```

5.6 NAREDBE ZA REALIZACIJU POTPROGRAMA

Ponekad rešavanje problema zahteva da se jedan isti postupak ponovi više puta. Program napisan za rešavanje takve vrste problema sadržao bi iste grupe naredbi na različitim mestima. Da bi se to preduzišlo, pomenuta grupa naredbi izdvaja se u posebnu celinu koja se naziva potprogram i na koju se može obraćati po potrebi. Na taj način moguće je ostvariti kraći zapis programa, kao i manje angažovanje memorije računara.

Pod potprogramom podrazumeva grupu naredbi u okviru programa koja se završava posebnom naredbom. Osnovne naredbe za realizaciju potprograma su GOSUB (obezbeđuje pozivanje potprograma) i RETURN (definiše poslednju naredbu u okviru potprograma).

GOSUB (Basic 2.0)

Obezbeđuje prelazak na potprogram. Opšti oblik naredbe je

GOSUB broj

pri čemu je

6*

broj — celobrojna vrednost koja određuje broj programske linije na koju se prenosi dalje izvršenje programa

Pri izvršavanju ove naredbe vrši se prelaz na programski red naznačen parametrom *broj*, koji ukazuje na prvu naredbu potprograma Nakon toga izvršavaju se naredbe počev od ove naredbe, pa sve dok se na nađe na progamsku liniju koja sadrži naredbu RETURN koja ukazuje da se došlo do kraja potprograma Dalje upravljanje se prenosi na naredbu koja sledi iza naredbe GOSUB.

Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

RETURN (Basic 2.0)

Služi za definisanje kraja potprograma, tj. obezbeđuje izlaz iz potprograma koji je pozvan naredbom GOSUB Opšti oblik naredbe je:

RETURN

Kada se pri izvršavanju potprograma nađe na naredbu RETURN, tada se vrši povratak u program, i to na naredbu koja sledi iza naredbe GOSUB Ova naredba se može koristiti isključivo u okviru potprograma, pri čemu svaki potprogram mora imati bar jednu RETURN naredbu.

Primeri

Primer 1 — Korišćenje potprograma

```
10 PRINT "RADI PROGRAM
20 GOSUB 100
30 PRINT "POVRATAK U PROGRAM"
40 PRINT "KRAJ PROGRAMA"
50 STOP
100 REM
101 REM POTPROGRAM
102 REM
110 PRINT "RADI POTPROGRAM"
120 RETURN
```

Primer 2 — Korišćenje potprograma sa više ulaza

Izračunati funkciju f(x) za zadatu vrednost argumenta *x*, pri čemu je:

$$f(x)=\begin{cases}\sin(x), & \text{za } x<0\\ \sqrt{x}, & \text{za } x\geqslant 0\end{cases}$$

```
10 REM
11 REM PROGRAM
12 REM
```

```
20 INPUT "ARGUMENT X=",X
30 IF X<0 THEN 90 ELSE 110
40 PRINT "X=",X: PRINT "F(X)=",Y
50 STOP
60 REM
70 REM PROGRAM ZA SRACUNAVANJE FUNKCIJE
80 REM
90 Y=SIN(X)
100 RETURN
110 Y=SQR(X)
120 RETURN
```

Program čine programske linije 10—50, a potprogram linije 60—120. U okviru programa prihvata se vrednost argumenta i na osnovu te vrednosti se određuje ulazna tačka u potprogram (programska linija 90 ili 110). Nakon ulaska u potprogram sračunava se vrednost zadate funkcije na jedan od dva načina (u zavisnosti od ulazne tačke), a zatim se dobijena vrednost predaje programu koji na ekranu prikazuje rezultat.

5.7. NAREDBE ZA REALIZACIJU PROGRAMSKIH PETLJI

Pod programskom petljom podrazumevamo grupu naredbi čije izvršenje treba ponavljati:

— određeni broj puta, ili
— sve dok se ne realizuje postavljeni uslov, ili
— sve dok je realizovan postavljeni uslov.

U prvom slučaju, za realizaciju programske petlje koriste se naredbe FOR i NEXT, i to respektivno kao prva i poslednja naredba programske petlje. Pri tome se uvodi promenljiva koja služi kao brojač realizovanih ciklusa.

U druga dva slučaja za realizaciju programske petlje koriste se naredbe DO, LOOP i EXIT. U ovom slučaju programska petlja je omeđena naredbom DO (početak petlje) i LOOP (kraj petlje), pri čemu se petlja ponavlja sve dok se ne realizuje zadati uslov (ili sve dok je realizovan postavljeni uslov), što se najčešće naznačava sa UNTIL ili WHILE. Ponekad se izlazni kriterijum iz petlje ostvaruje u kombinaciji naredbi IF i EXIT.

FOR (Basic 2.0)

Koristi se kao prva naredba programskog ciklusa koja treba izvršiti određeni broj puta. Opšti oblik naredbe je

FOR promenljiva = početak TO kraj [STEP korak]

pri čemu je

promenljiva — brojna promenljiva koja se naziva indeks ciklusa i služi kao kontrolna promenljiva pri izvršenju programske petlje,

početak — brojni izraz čija vrednost definiše početnu vrednost indeksa ciklusa,

kraj — brojni izraz čija vrednost definiše krajnju vrednost indeksa ciklusa,

korak — brojni izraz čija vrednost definiše priraštaj indeksa ciklusa (opcionog je karaktera i ukoliko se izostavi podrazumeva se da ima vrednost 1)

Naredbe koje slede iza naredbe FOR, zaključno sa naredbom NEXT, čine programski ciklus koji se ponavlja *n* puta, pri čemu je *n* celobrojna vrednost izraza

$$\frac{\text{kraj} - \text{početak}}{\text{korak}} + 1$$

Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

NEXT (Basic 2.0)

Koristi se kao poslednja naredba programskog ciklusa čiji je početak naznačen naredbom FOR, a koji treba izvršiti naznačeni broj puta. Opšti oblik naredbe je

NEXT [promenljiva]

pri čemu parametar *promenljiva* ima isto značenje kao u slučaju naredbe FOR. Generalno, programski ciklus ima strukturu

```
FOR I=P1 TO P2 STEP P3
.
.
naredbe BASIC jezika
.
.
NEXT I
```

i realizuje se na sledeći način. Najpre se indeksu ciklusa (promenljiva I) dodeljuje početna vrednost P1 i prelazi se na izvršavanje naredbi sadržanih u ciklusu — u redosledu u kome su naznačene. Nakon toga se indeks ciklusa uvećava za priraštaj P3, tj. dobija se vrednost P1+P3, i ponovo se izvršavaju naredbe sadržane u ciklusu. Prethodni postupak se ponavlja sve dok promenljiva I ne zadovolji uslov I≥P2. U tom trenutku se programski ciklus smatra završenim.

Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

DO (Basic 7.0)

Koristi se kao prva naredba programskog ciklusa koji se izvršava sve dok se ne zadovolji određeni uslov (ili sve dok je zadovoljan određeni uslov). Opšti oblik naredbe je

$$\text{DO} \quad \left[\begin{Bmatrix} \text{UNTIL} \\ \text{WHILE} \end{Bmatrix} \text{uslov} \right]$$

pri čemu je *uslov* — parametar sa istim značenjem kao u slučaju **IF** naredbe

Fraza UNTIL uslov se specificira ukoliko je petlju potrebno ponavljati sve dok se ne zadovolji određeni uslov, za razliku od fraze *WHILE uslov* koja se specificira kada petlju treba ponavljati sve dok je zadovoljen određeni uslov

Obe fraze su opcionog karaktera, i ukoliko se izostave — tada se izlazni kriterijum definiše ili u okviru naredbe **LOOP** (vidi naredbu LOOP) ili negde u okviru same petlje (korišćenjem IF naredbe u kombinaciji sa naredbom EXIT, o čemu će kasnije biti reči).

Naredbe koje slede naredbu DO, zaključno sa naredbom LOOP, definišu programsku petlju koja se izvršava sve dok se ne realizuje (ili sve dok je realizovan) određeni uslov

Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa

LOOP (Basic 7.0)

Služi kao poslednja naredba programskog ciklusa čiji je početak naznačen naredbom DO a koji treba izvršavati sve dok se ne realizuje (ili sve dok nije realizovan) naznačeni uslov. Opšti oblik naredbe je.

$$\text{LOOP} \quad \left[\begin{Bmatrix} \text{UNTIL} \\ \text{WHILE} \end{Bmatrix} \text{uslov} \right]$$

pri čemu parametar *uslov* i fraze *UNTIL uslov* i *WHILE uslov* imaju isto značenje kao u slučaju naredbe DO. Generalno, programski ciklus ovog tipa ima strukturu:

$$\text{DO} \left[\begin{Bmatrix} \text{UNTIL} \\ \text{WHILE} \end{Bmatrix} \text{uslov} \right]$$

naredbe

$$\text{LOOP} \left[\begin{Bmatrix} \text{UNTIL} \\ \text{WHILE} \end{Bmatrix} \text{uslov} \right]$$

pri čemu se u okviru ciklusa može pojaviti naredba EXIT. Ova na redba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

EXIT (Basic 7.0)

Koristi se kao naredba za izlaz iz programskog ciklusa definisanog naredbama DO i LOOP. Opšti oblik naredbe je:

EXIT

Pri realizaciji ove naredbe prekida se dalje izvršenje naredbi u okviru petlje tipa DO/LOOP i izvršenje programa se prenosi na prvu naredbu koja sledi iza naredbe LOOP. Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Primeri

Primer 1 — Korišćenje programske petlje tipa FOR/NEXT za prikazivanje neparnih brojeva iz opsega [1,99]

```
10 FOR I=1 TO 99 STEP 2
20 PRINT I : NEXT I
30 STOP
```

Primer 2 — Korišćenje programske petlje tipa DO/LOOP za prikazivanje neparnih brojeva iz opsega [1,99]

```
10 I=1
20 DO WHILE I<100
30 PRINT I : I=I+2
40 LOOP
```

U programu se najpre promenljivoj I dodeljuje vrednost 1, a zatim započinje realizacija petlje koju čine programske linije 20—40. Ova petlja se ponavlja sve dok je zadovoljen uslov I<100, pri čemu se u svakom prolazu kroz petlju na ekranu prikazuje odgovarajuća vrednost promenljive I, a nakon toga se ista uvećava za 2.

U ovom programu izlazni kriterijum iz petlje naznačen je u okviru naredbe DO. Ukoliko bismo želeli da isti zadatak realizujemo navođenjem izlaznog kriterijuma u okviru naredbe LOOP, tada bi program imao zapis

```
10 I=1
20 DO
30 PRINT I : I=I+2
40 LOOP WHILE I<100
```

Isti zadatak, rešen korišćenjem naredbe EXIT, ima oblik

```
10 I=1
20 DO
30 IF I<99 THEN EXIT
40 PRINT I; I=I+2
50 LOOP
```

Primer 3 — Korišćenje petlje tipa DO/LOOP

Napisati program kojim će se prihvatiti najviše 10 znakovnih podataka, pri čemu broj unetih podataka može biti i manji ukoliko je unet podatak KRAJ

```
10 DO UNTIL B=10: B=B+1
20 INPUT "PODATAK:" P$
30 IF P$="KRAJ" THEN EXIT
40 LOOP
50 PRINT "KRAJ PROGRAMA"
```

5.8. NAREDBE/KOMANDE ZA TESTIRANJE PROGRAMA I OBRADU GREŠAKA

Ovu grupu čine naredbe/komande HELP, TRAP, RESUME, TRON i TROFF. One predstavljaju novinu u Basic-u 7.0 i veoma su efikasno sredstvo kako pri pisanju i testiranju programa, tako i pri opisivanju postupaka za obradu izuzetih slučajeva. Zahvaljujući ovoj grupi naredbi/komandi znatno je olakšano pisanje većih programa. Ovde se pre svega misli na fazu testiranja programa.

HELP (Basic 7.0)

Koristi se za prikazivanje programske linije u kojoj je otkrivena greška u toku izvršavanja programa. Opšti oblik komande je

HELP

U direktnom načinu rada komanda se može aktivirati bilo pomoću tastera, HELP, bilo upisivanjem HELP na tastaturi.

Praktično se ova komanda koristi na sledeći način. Pretpostavimo da se izvršavao neki program i da je u toku izvršenja nastupila greška. O tome se na ekranu prikazuje odgovarajuća poruka. Da bismo videli sadržaj odgovarajuće programske linije (u kojoj je nastupila greška) potrebno je pritisnuti taster HELP. S obzirom na to da u jednoj programskoj liniji može biti sadržano više naredbi, to će na ekranu posebno biti naglašena naredba u kojoj je otkrivena greška. U slučaju 40-to kolonskog ekrana pomenuta naredba je prikazana u reverznom zapisu, a na 80-to kolonskom ekranu ona je prikazana podvučeno.

HELP komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

TRAP (Basic 7.0)

Služi za definisanje postupaka u slučajevima kada je u toku izvršenja programa došlo do greške. Opšti oblik naredbe je

TRAP [broj]

pri čemu je

broj — celobrojna vrednost koja određuje broj neke programske linije u okviru programa.

Naredba TRAP se obično smešta na početak programa i ona definiše određenu programsku liniju na koju će se izvršiti skok u slučaju otkrivanja greške pri izvršavanju programa, s napomenom da pomenuta naredba otkriva većinu grešaka koje mogu nastupiti u programu. U ovo nisu uključene greške nastale u radu sa disketnom jedinicom kao i izvestan broj greški koje nastaju u INPUT operacijama (REDO FROM START i EXTRA IGNORED). Pri definiciji postupka u slučaju greške najčešće se koriste sistemske promenljive EL, ER i ERR$(ER).

U najvećem broju slučajeva, u kombinaciji sa naredbom TRAP koristi se naredba RESUME, kojom se omogućuje povratak u program i njegovo dalje izvršanje (vidi naredbu RESUME). Naredba TRAP se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

RESUME (Basic 7.0)

Omogućuje nastavak izvršenja programa nakon što je naredbom TRAP otkrivena greška u izvršenju programa. Opšti oblik naredbe je

RESUME $\left[\left\{\begin{matrix}\text{broj}\\\text{NEXT}\end{matrix}\right\}\right]$

pri čemu je

broj — celobrojna vrednost (ne može biti promenljiva) koja određuje broj neke programske linije u okviru programa.

Ova naredba je obično poslednja naredba u okviru dela programa koji definiše postupak za obradu greške. RESUME naredba ima sledeće dejstvo u zavisnosti od argumenta:

a) ako je argument izostavljen, tada se izvršenje programa nastavlja počev od programske linije u kojoj je otkrivena greška;

b) ako je naznačen argument *broj*, tada se izvršava skok na programsku liniju koju on određuje i počev od nje nastavlja se izvršenje programa;

c) ako je naveden argument NEXT, tada se izvršenje programa nastavlja počev od programske linije koja sledi naredbu u kojoj je otkrivena greška.

Naredba RESUME se može koristiti isključivo na nivou programa.

TRON (Basic 7.0)

Koristi se za aktiviranje môda u kome se pri izvršavanju programa prikazuju obeležja programskih linija kroz koje se prolazi. Opšti oblik komande je

TRON

Ova komanda se najčešće koristi pri testiranju (tzv. *debugging*) programa, jer je veoma efikasna za praćenje rada programa. Može se koristiti bilo u direktnom načinu rada bilo na nivou programa.

TROFF (Basic 7.0)

Služi za isključivanje môda za testiranje u kome se pri izvršavanju programa prikazuju obeležja programskih linija kroz koje se prolazi. Opšti oblik komande je

TROFF

Komanda TROFF vraća računar u normalni režim rada (u pogledu izvršavanja programa), tj. služi za isključivanje komande TRON. Ova komanda se može koristiti bilo u direktnom načinu rada bilo na nivou programa.

Primeri

Primer 1 — Korišćenje komande HELP

```
10 REM
20 REM KORISCENJE KOMANDE HELP
30 REM
40 PRINT CHR$(147)
50 PRINT "BROJ KVADRAT KUB"
52 PRINT "================": PRIN
55 FOR I=1 TO 10
58 PRINT I,TAB(7),I*I,TAB(12):I*I*I
60 NEXT I
```

Aktiviranjem programa na ekranu dobijamo

```
BROJ KVADRAT KUB
================
?SYNTAX ERROR IN 52
```

što ukazuje da smo načinili grešku u sintaksi. Da bismo na ekranu prikazali programsku liniju u kojoj je otkrivena greška, dovoljno je da pritisnemo taster HELP. Nakon toga dobićemo

```
52 PRINT "=================== PRIN
```

Druga naredba u navedenoj programskoj liniji je prikazana inverzno (ili podvučeno) i ukazuje da je u njoj načinjena greška. Ispravimo zapis PRIN u PRINT (što možemo i direktno preko sadržaja koji smo dobili komandom HELP) i ponovo aktivirajmo program.

Primer 2 — Korišćenje naredbi TRAP i RESUME

```
10 TRAP 28
12 REM
14 REM RECIPROCNA VREDNOST
16 REM
18 INPUT "BROJNI PODATAK" B
20 R=1/B
22 PRINT "BROJ "B
24 PRINT "RECIPROCNA VREDNOST "R
26 STOP
28
30 REM OBRADA GRESKE
32
34 PRINT
36 PRINT "==================="
38 PRINT "PORUKA "ERR$(ER)
40 HELP
42 PRINT "==================="
44 PRINT : PRINT
46 RESUME 12
```

Program omogućava da se za zadatu vrednost brojnog podatka odredi njegova recipročna vrednost. Na ekranu se ispisuju obe vrednosti i prekida izvršenje programa. Ukoliko pri izvršenju programa nastupi greška, tada će se automatski izvršiti skok na programsku liniju 28 (što je specificirano sa prvom linijom programa, tj. TRAP 28). Deo programa koji čine linije 28—46 služi za obradu greške. Ovim delom procedure na ekranu će se prikazati odgovarajuća poruka o grešci, kao i odgovarajuća programska linija. Nakon toga završava se procedura za obradu greške i obavlja se skok na programsku liniju 12, što je realizovano sa RESUME 12.

Aktivirajmo program i kao brojni podatak upišimo 5. Dobićemo:

```
BROJ 5
RECIPROCNA VREDNOST 0.2
```

Ponovo aktivirajmo program, upisujući kao ulaz vrednost 0. Na ekranu se dobija

```
===================
PORUKA DIVISION BY ZERO
20 R=1/B
===================
BROJNI PODATAK ?
```

Naime, u ovom slučaju nastupila je greška u izvršavanju programa. Poruka DIVISION BY ZERO ukazuje da je došlo do pokušaja deljenja nulom, i to u programskom redu

20 R=1/B

S obzirom na proceduru koju smo opisali za obradu greške, program se ponovo aktivira počev od programskog reda 12. Ovo ima za posledicu poruku

BROJNI PODATAK 2

Upišimo, na primer, 12. Program će korektno nastaviti sa radom.

Primer 3 — Korišćenje komandi TRON i TROFF

Izvršimo izmene u prethodnom programu tako što ćemo dopisati sledeće programske redove:

11 TRON
25 TROFF

Aktiviranjem programa dobijamo:

[12][14][16][18] BROJNI PODATAK ?

Naime, u ovom slučaju se na ekranu unutar srednjih zagrada pojavljuju brojne vrednosti koje predstavljaju obeležja naredbi koje se izvršavaju. Pri tome niz pomenutih brojnih vrednosti definiše redosled izvršavanja programskih linija. Upišimo, na primer, kao ulaznu veličinu 4. Na ekranu dobijamo

[20][22] BROJ 4
[24] RECIPROČNA VREDNOST: 0.25
[25]
BREAK IN 26
READY

Prethodni primer je imao za cilj da pokaže kako se korišćenjem komandi TRON i TROFF može pratiti izvršenje programa, pri čemu se izloženo razmatranje može sprovesti na bilo kom delu programa. Ovo je svakako jedna od veoma značajnih pogodnosti, po sebno u slučaju komplikovanijih programa.

5.9 KOMANDE ZA RAD SA DISKETNOM JEDINICOM/JEDINICOM TRAKE

Ove komande se koriste za izvršavanje osnovnih operacija sa jedinicom diskete, odnosno trake. Ovoj grupi pripadaju komande HEADER, DIRECTORY, CATALOG, SAVE, DSAVE, BSAVE, VERIFY, DVERIFY, LOAD, DLOAD, BLOAD, RENAME, COPY, BACKUP, SCRATCH, COLLECT, CONCAT i BOOT.

Izvestan broj pomenutih komandi dostupan je i na jedinici diskete i na jedinici trake, dok se jedan broj komandi odnosi isključivo na jedinicu diskete.

S druge strane, određeni broj komandi je raspoloživ samo u C128 režimu rada. O ovim osobinama biće reči u svakom konkretnom slučaju pojedinačno

HEADER (Basic 7.0)

Koristi se za formatizaciju diskete. Opšti oblik komande je

HEADER "naziv" [,Iidentifikacija][,Ddisk] [{ON / ,} Ujedinica]

pri čemu je

naziv — parametar koji definiše naziv diskete (može biti dužine do 16 karaktera),

identifikacija — parametar služi za identifikaciju diskete (može biti dužine do 2 karaktera i poželjno je da bude jedinstven za svaku disketu),

disk — parametar kojim se definiše vrednost dodeljena disku. Može imati vrednost 0 ili 1. Ukoliko se izostavi — podrazumeva se *D0*,

jedinica — parametar kojim se definiše vrednost dodeljena disketnoj jedinici. Dozvoljene su vrednosti 8 ili 9. Ukoliko se izostavi — podrazumeva se *U8*

Svaka disketa mora biti formatizovana pre upotrebe. U tu svrhu koristi se komanda HEADER. Njome se na disketi kreira tabela poznata pod nazivom "direktorijum", u koju se kasnije upisuju osnovne informacije o pojedinim datotekama

Komanda HEADER može biti korišćena bilo u direktnom načinu rada, bilo iz programa. Ako se koristi u direktnom načinu rada, tada se na ekranu pojavljuje pitanje

ARE YOU SURE ?

U zavisnosti od odgovora moguće je izvršiti formatizaciju ili od nje odustati. Ova osobina je značajna s obzirom na to da se formatizacijom diskete uništava njen prethodni sadržaj (ukoliko postoji)

Primer 1 — HEADER "RADNA", I01

Ovom komandom izvršava se formatizacija diskete pod nazivom RADNA sa identifikacijom 01. Pri radu se podrazumeva da se disketa nalazi na disku 0 (u komandi je podrazumevana vrednost D0) u jedinici 8 (podrazumevan parametar U8)

DIRECTORY (Basic 7.0)

Koristi se za prikazivanje sadržaja (čitavog ili dela) disk-direktorija na ekran. Opšti oblik komande je

DIRECTORY [Ddisk] [{ON / ,} Ujedinica]['slika']

pri čemu je

disk — parametar sa istim značenjem kao u slučaju komande HEADER,

jedinica — parametar sa istim značenjem kao u slučaju komande HEADER.

slika — parametar kojim se definiše deo sadržaja disk-direktorija koji će biti prikazan na ekranu. U tu svrhu se pored standardnih simbola koji učestvuju u formiranju naziva datoteka koriste i simboli * i ?. O ovome će biti više reči u primerima koji slede. Parametar *slika* je opcionog karaktera i ukoliko se izostavi — podrazumeva se prikazivanje kompletnog sadržaja direktorija.

Nakon aktiviranja komande DIRECTORY na ekranu se prikazuje odgovarajući sadržaj direktorija, pri čemu program smešten u memoriji računara ostaje nepromenjen. U prikazu koji se dobija na ekranu sadržane su osnovne informacije o disketi i datotekama na njoj. U vrhu ekrana naznačen je naziv diskete i odgovarajuća identifikacija (vidi komandu HEADER), a ispod toga nazivi datoteka. S leve strane naziva svake od datoteka prikazan je broj koji ukazuje na veličinu prostora koji ona zauzima a s desne strane odgovarajući tip datoteke.

Ova komanda se može aktivirati i pomoću funkcionalnog tastera F3. Zaustavljanje listanja može se realizovati tasterom NO SCROLL (ili CTRL-S). Komanda DIRECTORY se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Napomena. Ova komanda se ne može koristiti za listanja direktorija na štampaču. U tu svrhu koristimo

```
LOAD '$0'8
OPEN 4 4 CMD4 LIST
PRINT#4 CLOSE4
```

SAVE (Basic 2.0)

Koristi se za upisivanje programa, smeštenog u memoriji računara na spoljašnju jedinicu (kasetu ili disketu). Opšti oblik komande je

SAVE ["naziv"][,jedinica][,adresa]

pri čemu je

naziv — naziv pod kojim će program biti upisan na eksternu jedinicu (može biti izostavljen u slučaju zapisa na kasetu),

jedinica — opcioni parametar koji specificira eksternu jedinicu. Ukoliko se izostavi, podrazumeva se vrednost 1. Moguće vrednosti su
1 — jedinica kasete,
8 — jedinica diskete,

adresa — opcioni parametar koji definiše način zapisa. Ukoliko se izostavi, podrazumeva se *blanko*. Moguće vrednosti su:

blanko — podrazumeva se takav zapis koji se kasnijim punjenjem u memoriju računara smešta na standardni način,

1 — zapis koji pri kasnijem učitavanju programa u memoriju računara treba smestiti na iste lokacije sa kojih je izvršen zapis,

2 — ova vrednost se koristi isključivo za slučaj zapisa na traku. Pri tome se na kraju programa upisuje oznaka koja ukazuje na kraj trake.

3 — ova vrednost predstavlja kombinaciju vrednosti 1 i 2.

Primer 1 — SAVE "PROGRAM"

Ovom komandom se na traku upisuje program pod nazivom PROGRAM.

Primer 2 — SAVE "IGRA",8

Ovom komandom se na disketu upisuje program pod nazivom IGRA.

Primer 1 — DIRECTORY

Ova komanda se koristi za prikazivanje kompletnog sadržaja disk-direktorija. Podrazumeva se disk 0 (tj. D0) i jedinica 8 (tj. U8).

Primer 2 — DIRECTORY D1, U9, "DAT*"

Ova komanda se koristi za prikazivanje svih datoteka čiji nazivi počinju slovima DAT (na primer DAT1, DATOTEKA, DATUM ...). Pri tome je disketa na disku 1 (korišćen je parametar D1) u jedinici 9 (parametar U9).

Primer 3 — DIRECTORY "D?K"

Ovom komandom se na ekranu prikazuju informacije o svim datotekama u čijem je nazivu prvo slovo D, poslednje K, a na drugoj poziciji se nalazi bilo koji karakter (na primer DAK, DEK, DOK, DSK ...). Pri tom se podrazumeva disk 0 i jedinica 8.

CATALOG (Basic 7.0)

Ima istu namenu kao i komanda DIRECTORY, tj. koristi se za prikazivanje sadržaja (čitavog ili dela) disk-direktorija. Opšti oblik komande je:

CATALOG [Ddisk] [{ON / ,} Ujedinica] [,"slika"]

pri čemu parametri *disk*, *jedinica* i *slika* imaju isto značenje kao u slučaju komande DIRECTORY. Za detalje o ovoj komandi videti komandu DIRECTORY.

DSAVE (Basic 7.0)

Koristi se za upisivanje programa, smeštenog u memoriji računara, na disketu. Opšti oblik komande je:

DSAVE "naziv" [,Ddisk] $\left[\begin{Bmatrix} ON \\ , \end{Bmatrix}\right.$ Ujedinica $\left.\right]$

pri čemu je:

naziv — naziv pod kojim će program biti upisan na disketu. Maksimalna dužina može biti 16 karaktera.

Parametri disk i *jedinica* imaju isto značenje kao u slučaju komande HEADER.

Primer 1 — DSAVE "PROG"

Ovom komandom se na disketu upisuje program pod nazivom PROG. Pri tome se podrazumeva disk 0 i jedinica 8.

Primer 2 — DSAVE (A$)

Ovom komandom se na disketu upisuje program pod nazivom koji je određen sadržajem promenljive A$. Kao u prethodnom primeru — podrazumeva se disk 0 i jedinica 8.

Komanda DSAVE se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

BSAVE (Basic 7.0)

Služi za upisivanje bloka memorije (iz bilo koje banke podataka) na disketu. Opšti oblik komande je:

BSAVE "naziv" [,Ddisk] $\left[\begin{Bmatrix} ON \\ , \end{Bmatrix}\right.$ Ujedinica $\left.\right]$
[,Bbanka] Početak TO Pkraj

pri čemu je

naziv — naziv pod kojim će blok memorije biti upisan na disketu. Može biti dužine do 16 karaktera,

banka — celobrojna vrednost kojom se definiše jedna od 16 mogućih standardnih memorijskih konfiguracija (videti komandu BANK u sledećem poglavlju),

početak — parametar kojim se definiše početna lokacija bloka memorije čiji sadržaj želimo da upišemo na disketu,

kraj — parametar kojim se definiše krajnja lokacija bloka memorije čiji sadržaj želimo da upišemo na disketu

Parametri *disk* i *jedinica* imaju isto značenje kao u slučaju komande HEADER

Za razliku od komanda SAVE i DSAVE, koje služe za upisivanje BASIC programa na disketu, komandom BSAVE se najčešće upisuju mašinski programi, podaci za sprajtove i sadržaj ekrana u grafici visoka rezolucije, kao i sadržaji drugih delova memorije koji su izvan zone namenjene BASIC programima Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Primer 1 — BSAVE "SPRAJT",B0,P3584 T OP4096

Ovom komandom će na disketi, pod nazivom SPRAJT biti upisan blok memorije počev od lokacije 3584 pa sve do lokacije 4095. Pri tom se podrazumeva da je blok smešten u banci 0 (što je određeno parametrom B0)

VERIFY (Basic 2 0)

Koristi se za upoređivanje BASIC programa, smeštenog na traci ili disketi, sa programom koji se trenutno nalazi u memoriji računara Opšti oblik komande je

VERIFY ["nsov"][jedinica][,adresa]

pri čemu je:

naziv — naziv programa na traci/disketi čiji sadržaj treba uporediti sa sadržajem programa koji se nalazi u memoriji računara. Ukoliko se izostavi — podrazumeva se tekući program na traci,

jedinica — opcioni parametar koji može imati vrednost 1 ili 8 — u zavisnosti od toga da li je reč o programu na traci ili disketi Ukoliko se izostavi — podrazumeva se jedinica trake,

adresa — parametar koji definiše na koji se način program učitava u memoriju računara (vidi komandu LOAD)

Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa

Primer 1 — VERIFY "PROG1"

Ovom komandom vrši se upoređivanje programa smeštenog u memoriji računara sa programom koji je zapisan na traci pod nazivom PROG1

DVERIFY (Basic 7.0)

Koristi se za upoređivanje BASIC programa, smeštenog na disketi, sa programom koji se trenutno nalazi u memoriji računara. Opšti oblik komande je

DVERIFY "naziv" [,Ddisk] $\left[\left\{\begin{matrix}ON\\,\end{matrix}\right\}\text{ Ujedinica}\right]$

pri čemu je:

naziv — naziv programa na disketi čiji sadržaj treba uporediti sa programom koji se trenutno nalazi u memoriji računara.

Parametri *disk* i *jedinica* imaju isto značenje kao u slučaju komande HEADER.

Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Primer 1 — DVERIFY "TEST"

Ovom komandom vrši se upoređivanje programa smeštenog na disketi pod nazivom TEST — sa programom koji se trenutno nalazi u memoriji računara. Pri tome se podrazumevaju — disk 0 i jedinica 8

LOAD (Basic 2.0)

Koristi se za čitanje programa sa eksterne jedinice (kaseta ili diskete) i njegovo smeštanje u memoriju računara. Opšti oblik komande je

LOAD ["naziv"][,jedinica][,adresa]

pri čemu je

naziv — znakovni podatak koji predstavlja naziv programa koji treba učitati u memoriju računara. Opcionog je karaktera u slučaju da je program smešten na kaseti.

jedinica — celobrojna vrednost koja definiše jedinicu sa koje treba izvršiti učitavanje naznačenog programa. Moguće vrednosti su:

1 — jedinica trake,
8 — jedinica diskete

Ukoliko se parametar izostavi, podrazumeva se jedinica trake.

adresa — celobrojna vrednost koja definiše način smeštanja programa u memoriju računara. Moguće vrednosti su

0 — program se smešta u memoriju računara na standardan način,
1 — program se smešta u memoriju računara na iste lokacije sa kojih je i zapisan komandom SAVE. Ukoliko se parametar izostavi podrazumeva se vrednost 0

Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Primer 1 — LOAD

Ovom komandom se učitava u memoriju prvi program sa trake (u odnosu na tekuću poziciju glave za čitanje)

Primer 2 — LOAD "MASINAC",8,1

Ovom komandom se sa diskete učitava program pod nazivom MASINAC i smešta se u memoriju računara na iste lokacije sa kojih je i zapisan komandom SAVE.

DLOAD (Basic 7.0)

Koristi se za čitanje programa sa disketne jedinice i njegovo smeštanje u memoriju računara. Opšti oblik komande je

DLOAD "naziv" [,Ddisk] [{ON / ,} Ujedinica]

pri čemu je:

naziv — znakovni podatak koji predstavlja naziv programa koji treba učitati u memoriju računara.

Parametri *disk* i *jedinica* imaju isto značenje kao u slučaju komande HEADER. Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Napomena. U okviru naziva programa koji treba učitati sa diskete mogu se koristiti simboli * i ? koji imaju specijalno značenje. Ovu osobinu ćemo koristiti u primerima koji slede.

Primer 1 — DLOAD "AS*"

Ovom komandom se sa diskete učitava prvi program iz direktorija čije ime počinje sa AS. Pri tom se podrazumeva disk 0 i jedinica 8.

Primer 2 — DLOAD "PROG?1",D1 ON U9

Ovom komandom se sa diskete učitava program čija su prva četiri karaktera u nazivu PROG, poslednji karakter cifra 1, a peti karakter bilo koji simbol. Pri tom se disketa nalazi u disku čiji su parametri D1 i U9.

BLOAD (Basic 7.0)

Služi za čitanje bloka podataka sa diskete (koji je zapisan komandom BSAVE ili pomoću mašinskog monitor programa) i njegovo smeštanje u memoriju računara počev od naznačene memorijske lokacije pri čemu je naznačena i određena memorijska konfiguracija. Opšti oblik komande je

BLOAD "naziv" [,Ddisk] [{ON / ,} Ujedinica]
[,Bbanka][,Ppočetak]

pri čemu parametri *naziv, disk, jedinica, banka* i *početak* imaju isto značenje kao u slučaju komande BSAVE.

Komande LOAD i DLOAD služe za učitavanje BASIC programa u memoriju računara. S druge strane, BLOAD se koristi za učitavanje mašinskih programa, definicija za sprajtove, definicija za funkcionalne ključeve i sl. Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Primer 1 — BLOAD "SPRAJT",D0,U8,B0,P3584

Ova komanda se koristi za učitavanje datoteke SPRAJT i njeno smeštanje u memoriju računara počev od lokacije 3584 u banci 0. Pri tome je datoteka sa diska 0 i jedinice 8.

RENAME (Basic 7.0)

Koristi se za promenu naziva datoteke smeštene na disketi. Opšti oblik komande je:

RENAME "naziv1" TO "naziv2" [,Ddisk] $\left[\left\{\begin{matrix}ON\\,\end{matrix}\right\}\right.$ U*jedinica* $\left.\right]$

pri čemu je

naziv1 — naziv neke od postojećih dadoteka na disketi, čiji naziv treba promeniti,

naziv2 — novi naziv datoteke.

Parametri *disk* i *jedinica* imaju isto značenje kao u slučaju komande HEADER.

Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Primer 1 — RENAME "TEST1" TO "TEST2"

Ovom komandom je naziv datoteke TEST1 promenjen u naziv TEST2. Pri tome se podrazumeva disk 0 i jedinica 8.

COPY (Basic 7.0)

Koristi se za kopiranje (prepisivanje) sadržaja jedne datoteke u drugu. Opšti oblik komande je:

COPY ["naziv1"] [,Ddisk1] TO ["naziv2"] [,Ddisk2] $\left[\left\{\begin{matrix}ON\\,\end{matrix}\right\}\right.$ U*jedinica* $\left.\right]$

pri čemu je

naziv1 — naziv datoteke (na disku *disk1*) koju treba kopirati,

naziv2 — naziv datoteke (na disku *disk2*) koja se dobija nakon kopiranja

Parametri *disk1, disk2* i *jedinica* imaju isto značenje kao u slučaju komande HEADER.

Ova komanda se može koristiti za:

— kopiranje jedne datoteke sa jednog diska u datoteku na drugom disku (reč je o dvostrukoj disk jedinici),

— kopiranje jedne datoteke sa jednog diska u drugu datoteku na istom disku

— kopiranje svih datoteka sa jednog diska na drugi disk (reč je o dvostrukoj disk jedinici)

Napomena. Komandom COPY nije moguće ostvariti kopiranje između dve:

a) jednostruke *(single)* disketne jedinice,

b) dvostruke *(double)* disketne jedinice

Pomenuta komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa

Primer 1 — COPY "DAT1" T0 "DAT2"

Komanda za kopiranje datoteke DAT1 u datoteku DAT2, pri čemu se podrazumeva da su obe datoteke na disku 0 i jedinici 8.

Primer 2 — COPY "DAT",D0 T0 "DAT",D1

Komanda za kopiranje datoteke DAT sa diska 0 na datoteku sa istim imenom na disku 1. Pri tom se podrazumeva da su oba diska u istoj disketnoj jedinici 8

Primer 3 — COPY D0 T0 D1

Ovom komandom se kopiraju sve datoteke sa diska 0 na disk 1. Pri tom su oba diska u istoj disketnoj jedinici

BACKUP (Basic 7.0)

Koristi se za kopiranje čitavog sadržaja jedne diskete na drugu. Opšti oblik komande je

BACKUP Ddisk1 T0 Ddisk2 $\left[\begin{Bmatrix} ON \\ , \end{Bmatrix} \text{Ujedinica}\right]$

pri čemu je:

disk1 — parametar koji definiše disk sa koga se vrši kopiranje,

disk2 — parametar koji definiše disk na koji se vrši kopiranje

Parametar *jedinica* ima isto značenje kao u slučaju komande HEADER

Napomene.

a) Ova komanda se može koristiti isključivo na dvostrukoj disketnoj jedinici, kao što je jedinica 1572.

b) Disketa na koju se vrši kopiranje ne mora biti formatizovana, s obzirom na to da se kompletan sadržaj diskete sa diska *disk1* prepisuje na disketu koja je na disku *disk2*

Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Primer 1 — BACKUP D0 TO D1

Ovom komandom se kopira kompletna disketa sa diska *disk1* na disketu na disku *disk2*. Podrazumeva se da je reč o dvostrukoj disketnoj jedinici čija je identifikacija 8

SCRATCH (Basic 7.0)

Koristi se za brisanje određene datoteke na disketi. Opšti oblik komande je

SCRATCH "naziv" [,Ddisk] [{ON / ,} Ujedinica]

pri čemu je:

naziv — parametar koji definiše naziv datoteke koju treba obrisati

Parametri *disk* i *jedinica* imaju isto značenje kao u slučaju komande HEADER

Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa. U direktnom načinu rada, nakon aktiviranja komande pojavljuje se pitanje

ARE YOU SURE ?

što omogućava da se operacija izvrši ili da se od nje odustane — u zavisnosti od odgovora.

Primer 1 — SCRATCH "PROBA",D0

Ovom komandom se vrši brisanje datoteke sa nazivom PROBA, koja je smeštena na disketi u disku 0

COLLECT (Basic 7.0)

Koristi se za oslobađanje prostora na disku koji je zauzet nekorektno zatvorenim datotekama (datoteke koje iza naziva imaju simbol *).

Opšti oblik komande je:

COLLECT [Ddisk] [{ON / ,} Ujedinica]

pri čemu parametri *disk* i *jedinica* imaju isto značenje kao u slučaju komande HEADER. Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa

Primer 1 — COLLECT D0

Ovom komandom se oslobađa prostor na disketi (smeštenoj u disku 0) koji je zauzet nekorektno zatvorenim datotekama.

CONCAT (Basic 7.0)

Služi za povezivanje (spajanje) dve datoteke u jednu. Opšti oblik komande je

CONCAT "naziv1" [,Ddisk1] TO "naziv2" [,Ddisk2] [{ON / ,} Ujedinica]

pri čemu je

naziv1 — parametar koji definiše naziv datoteke na disketi u disku koji je određen parametrom *disk1*,

naziv2 — parametar koji definiše naziv datoteke na disketi u disku koji je određen parametrom *disk2*

Parametri *disk1*, *disk2* i *jedinica* imaju isto značenje kao u slučaju komande HEADER

Komandom CONCAT se sadržaj datoteke *naziv1* dodaje na kraj datoteke *naziv2*, pri čemu novodobijena datoteka postaje *naziv2*. U pomenutoj operaciji sadržaj datoteke *naziv1* ostaje nepromenjen. Ova komanda se odnosi isključivo na datoteke smeštene na disketi i može se koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Primer 1 — CONCAT "DATA" TO "DATB"

Navedena komanda omogućuje promenu sadržaja datoteke DATB na taj način što će na kraj ove datoteke biti dodat(dopisan) sadržaj datoteke DATA. Nakon izvršene komande sadržaj datoteke DATA ostaje nepromenjen

BOOT (Basic 7.0)

Koristi se za učitavanje u memoriju i izvršavanje mašinskog programa smeštenog na *autoboot disketi* (disketa koja na određenoj lokaciji sadrži informacije o automatskom aktiviranju programa). Opšti oblik komande je

BOOT ["naziv"] [,Ddisk] [{ON / ,} Ujedinica]

pri čemu je

naziv — parametar koji definiše naziv programa koji je smešten na disketi

Parametri *disk* i *jedinica* imaju isto značenje kao u slučaju komande HEADER

Nakon uključivanja računara aktivira se Kernal-ova rutina BOOT-CALL, koja pokušava da učita sadržaj određene lokacije sa diskete (trag 1, sektor 0). Ukoliko pri tome disketna jedinica nije priključena ili u njoj nije postavljena disketa ili na odgovarajućoj lokaciji diskete ne postoji autobot informacija — tada rutina BOOT-CALL prekida sa radom, a kontrola se prenosi na BASIC. U slučaju da su ispunjeni pomenuti uslovi, pronalazi se odgovarajući program, učitava se u memoriju računara i aktivira se, o čemu se na ekranu dobija odgovarajuća informacija. Ovo je tzv *automatski BOOT* koji se izvršava bez dejstva korisnika

Ako se u toku rada računara aktivira komanda BOOT (bez parametara), tada se izvršava postupak analogan automatskom BOOT-u, ukoliko je uključena disketa koja ima "autoboot" informaciju

Ukoliko se u toku rada aktivira komanda BOOT sa navedenim parametrima, tada se na disketi pronalazi odgovarajući program, učitava se u memoriju i aktivira se

Ova komanda se može aktivirati bilo u direktnom načinu rada, bilo na nivou programa.

Primer 1 — BOOT "POCETAK"

Ovom komandom se sa diskete učitava program sa nazivom POCETAK u memoriju računara a nakon toga se aktivira. Pri tom se podrazumeva disk 0 i jedinica 8

5.10 SISTEMSKE NAREDBE/KOMANDE

Ovoj grupi pripadaju naredbe/komande: BANK, POKE, G064, FAST, SLOW, SYS, FETCH, STASH i SWAP

BANK (Basic 7.0)

Koristi se za izbor jedne od 16 memorijskih konfiguracija koje su numerisane brojevima 0—15. Opšti oblik komande je

BANK broj

pri čemu je
broj — celobrojna vrednost iz intervala [0,15], od kojih svaka definiše jednu memorijsku konfiguraciju (tzv "banku").

Računar C-128 raspolaže sa 128K RAM memorije i 52K ROM memorije. S obzirom na to da se u svakom trenutku može adresirati samo 64K (tj 65536 bajtova) napravljeno je 16 različitih "pogleda"

na memoriju. Svaki od njih definiše jednu memorijsku konfiguraciju, tj. banku memorije. Tako, na primer, banka 0 obuhvata onaj deo RAM memorije koji se koristi za smeštanje BASIC programa, a banka 1 deo RAM memorije namenjen za definiciju i smeštaj podataka koji se koriste u BASIC programima.

Komanda BANK služi za postavljanje pokazivača na odgovarajući deo memorije koji treba učiniti aktivnim za naredbe PEEK, POKE, SYS i ostale koje u sebi sadrže parametre koji predstavljaju adrese u memoriji. Ova komanda se može koristiti bilo u direktnom načinu rada bilo na nivou programa.

Primer 1 — Korišćenje komande BANK

```
10 BANK 1
12 A%=27  P=POINTER(A%)
14 FOR I=P-2 TO P+4
16 PRINT I PEEK(I)
18 NEXT I
20 PRINT
22 PRINT "A%=",A%
24
26 REM
28 REM RACUN
30 REM
32 PRINT "NAZIV PROMENLJIVE ";
34 PRINT CHR$(PEEK(P-2)—PEEK(P-1))
36 PRINT "VREDNOST PROMENLJIVE ",
38 PRINT PEEK(P)*256+PEEK(P+1)
```

Prva programska linija služi za postavljanje pokazivača na banku 1, koja je namenjena za definisanje i smeštaj promenljivih u BASIC programu. Naredni programski red služi za dodeljivanje vrednosti jednoj celobrojnoj promenljivoj i određivanje pointera za tu promenljivu (vidi funkciju POINTER).

Programski redovi 14—18 predstavljaju programsku petlju kojom se prikazuju lokacije (1024—1030) i sadržaji tih lokacija koji ukazuju na promenljivu A%. Pri tome prva dva bajta definišu naziv promenljive, a sledeća dva sadržaj promenljive A%. Poslednja tri bajta ostaju neiskorišćena.

Programske linije 26—38 omogućuju određivanje naziva promenljive i odgovarajuće vrednosti na osnovu pokazivača.

Napominjemo da smo se u ovom primeru ograničili na celobrojnu promenljivu.

Aktiviranjem prethodnog programa dobijamo rezultat

```
1024    193
1025    128
1026    0
1027    27
```

```
1028     0
1029     0
1030     0
A=27
NAZIV PROMENLJIVE  A
VREDNOST PROMENLJIVE  27
```

POKE (Basic 2.0)

Koristi se za smeštanje proizvoljne brojne vrednosti na određenu memorijsku lokaciju (u okviru tekuće banke). Opšti oblik naredbe je

POKE adresa,broj

pri čemu je:

adresa — brojna vrednost koja definiše memorijsku lokaciju na koju treba smestiti brojnu vrednost definisanu parametrom *broj*. Pri tome je $0 \leq adresa \leq 65535$.

broj — brojna vrednost koju treba smestiti na memorijsku lokaciju definisanu parametrom *adresa*. Pri tome je $0 \leq broj \leq 255$.

POKE naredba omogućava promenu sadržaja bilo koje adrese unutar RAM memorije, kao i većeg broja ulazno/izlaznih registara. Ova naredba se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Primer 1 — 10 BANK 0: POKE 16000,1

U ovom primeru se brojna vrednost 1 smešta na lokaciju 16000 u okviru banke 0.

GO64 (Basic 7.0)

Koristi se za prelaz iz C-128 u C-64 način rada. Opšti oblik komande je:

GO64

Može se koristiti bilo u direktnom načinu rada, bilo na nivou programa. Pri aktiviranju komande u direktnom načinu rada na ekranu se pojavljuje pitanje:

ARE YOU SURE?

U zavisnosti od odgovora moguće je preći u C-64 mód ili ostati u C-128 režimu rada.

Napomena. Ne postoji komanda za povratak iz C-64 u C-128 način rada. U tu svrhu treba jednostavno isključiti i ponovo uključiti računar.

FAST (Basic 7.0)

Služi za iniciranje mikroprocesora 8502 na brzinu od 2MHz. Opšti oblik komande je

FAST

Nakon aktiviranja komande FAST, brzina procesorskog sata postaje dva puta veća, tj. povećava se sa 1 na 2 miliona ciklusa u sekundi. Pri tome se takođe isključuje 40-to kolonski ekran (u slučaju 80-to kolonskog ekrana nema nikakvih promena).

Ova komanda je pogodna za programe u kojima ima dosta računanja — s obzirom na to da se značajno ubrzavaju sve operacije (sa izuzetkom ulazno/izlaznih operacija).

FAST način rada se kontroliše sadržajem lokacije 53296 u banci 15. Ako ova lokacija ima sadržaj 1, tada je uključen FAST mód, a u protivnom ja isključen. Ova komanda se može koristiti bilo u direktnom načinu rada, bilo na nivou programa.

Primer 1 — Rad u FAST módu

```
10 TI$="000000"
20 FAST
30 FOR I=1 TO 1000
40 A=A+1
50 SLOW
60 PRINT "RAD U FAST MODU ",
70 PRINT "TRAJE",TI/60;"SEC"
```

Za trenutak pomenimo da komanda SLOW omogućuje povratak u normalni način rada. Izvršenje ovog programa traje oko 2.7167 sec.

SLOW (Basic 7.0)

Koristi se za vraćanje mikroprocesora 8502 iz FAST načina rada na standardnu brzinu od 1MHz. Opšti oblik komande je

SLOW

Nakon aktiviranja komande SLOW, 40-to kolonski ekran (ili ekran visoke rezolucije) postaje raspoloživ. Ova komanda se može koristiti bilo u direktnom, bilo u odloženom načinu rada.

Primer 1 — Rad u SLOW módu

```
10 SLOW
20 TI$="000000"
30 FOR I=1 TO 1000
40 A=A+1
```

```
50 NEXT I
60 PRINT "RAD U SLOW MODU",
70 PRINT " TRAJE ',TI/60,' SEC"
```

Izvršenje ovog programa traje oko 5.6333 sec.

SYS (Basic 2.0)

Koristi se za aktiviranje izvršenja mašinskog programa koji je smešten u tekućoj banci, počev od naznačene adrese. Opšti oblik naredbe je:

SYS adresa

pri čemu je:

adresa — celobrojna vrednost koja može biti iz opsega [0,65535] i predstavlja adresu od koje je smešten mašinski program koji treba izvršiti.

SYS naredba omogućuje pozivanje mašinskog programa iz BASIC programa. Pri tome, pre pozivanja naredbe SYS treba u memoriju računara smestiti odgovarajući mašinski program. Ovo se obično realizuje komandom BLOAD ili grupom POKE naredbi.

Postoji određen broj mašinskih programa koji su ugrađeni u ROM-u računara koji se mogu pozvati naredbom SYS. Jedna od njih je:

BANK 15: SYS 65341

kojim se računar dovodi u početno stanje nakon uključivanja računara. Ova naredba se može aktivirati bilo u direktnom načinu rada, bilo na nivou programa.

Napomena: U Basic-u 7.0 izvršena je izmena naredbe SYS tako da ona ima opšti oblik:

SYS adresa [,a] [,x] [,y] [,s]

pri čemu su *a*, *x*, *y* i *s* opcioni parametri kojima se (ukoliko su navedeni) mogu definisati vrednosti koje će biti smeštene respektivno u akumulator, X-registar, Y-registar i status-registar.

Sledeće tri naredbe omogućuju razmenu podataka između memorije računara i RAM modula koji predstavlja proširenje RAM memorije računara. Ovo proširenje se može koristiti za smeštanje programa i/ili podataka. Naredbe koje to omogućuju su FETCH, STASH i SWAP.

FETCH (Basic 7.0)

Koristi se za prihvatanje podataka iz RAM modula u memoriju računara. Opšti oblik naredbe je:

FETCH broj, adresa1,adresa2,banka

pri čemu je

broj — celobrojna vrednost iz opsega [0,65535] koja predstavlja broj bajtova koji će biti prihvaćeni iz RAM modula,

adresa1 — celobrojna vrednost iz opsega [0,65535] koja određuje adresu u okviru memorije računara (u tekućoj banci) od koje počinje smeštanje podataka koji su prihvaćeni iz RAM modula,

adresa2 — celobrojna vrednost iz opsega [0,65535] koja određuje adresu u okviru određene banke RAM modula od koje počinju podaci koje treba prebaciti u memoriju računara,

banka — celobrojna vrednost iz opsega [0,15] koja definiše odgovarajuću banku u okviru RAM modula.

STASH (Basic 7.0)

Služi za smeštanje podataka iz memorije računara u RAM modul. Opšti oblik naredbe je

STASH broj,adresa1,adresa2,banka

pri čemu parametri *broj*, *adresa1*, *adresa2* i *banka* imaju isto značenje kao u slučaju naredbe FETCH.

SWAP (Basic 7.0)

Koristi se za zamenu sadržaja memorije računara i sadržaja RAM modula. Opšti oblik naredbe je:

SWAP broj,adresa1,adresa2,banka

pri čemu parametri **broj**, **adresa1**, **adresa2** i **banka** imaju isto značenje kao u slučaju naredbe FETCH.

5.11. GRAFIKA

U ovom poglavlju objašnjavamo grafičke naredbe koje podržava Basic 7.0. To su isključivo njegove naredbe, jer Basic 2.0 (Basic u modu 64) nema nijednu naredbu koja podržava rad sa grafikom. Pre nego počnemo sa objašnjavanjem naredbi, rećićemo nešto o grafičkim mogućnostima COMMODORA 128. Pošto se u kompjuteru nalaze dva čipa za generisanje slike, grafičke mogućnosti podeličemo u dve grupe.

Slika koju dobijamo preko televizora ili monitora sa kompozitnim ulazom ima dva različita moda rada: tekst mod od 40 karaktera u redu i grafički mod sa rezolucijom 320 tačaka po horizontali i 200 tačaka po vertikali u dve boje. Ako nam je potrebno više od dve boje na ekranu možemo koristiti i višebojni grafički mod. Kod njega imamo nešto slabiju rezoluciju od 160 tačaka po horizontali i 200 tačaka po vertikali u četiri boje. Kod prvog grafičkog moda jednoj tački na

ekranu odgovara jedan bit u memoriji. Bit može biti 0 ili 1, tj. tačka postoji ili ne postoji. Jasno nam je da u ovom grafičkom modu možemo imati samo dve boje. Kod višebojne grafike jednoj tački na ekranu odgovaraju dva bita u memoriji. Bitovi koji odgovaraju jednoj tački mogu biti sledeći: 00,01,10 i 11. Pošto imamo četiri različite kombinacije, jasno je da u tom grafičkom modu možemo ostvariti najviše četiri različite boje.

Ako sliku dobijamo preko RGB monitora (na televizoru ovu sliku ne možemo dobiti), imamo takođe dva različita moda rada: tekst mod od 80 karaktera u redu i 25 redova i grafički mod sa rezolucijom od 640 tačaka po horizontali i 200 tačaka po vertikali u dve boje. Moramo napomenuti da ovaj grafički mod ne podržavaju grafičke naredbe Basic-a 7.0. (Na kraju knjige u posebnom prilogu dat je mašinski program koji Basic-u 7.0 dodeljuje nove naredbe koje podržavaju ovaj grafički mod.)

Sada ćemo redom objasniti svaku grafičku naredbu, dati njenu sintaksu i ilustrovati kratkim primerom. Iza svake naredbe sledi lista parametara koji određuju tu naredbu. U sledećem poglavlju daćemo duže primere — kako biste bolje shvatili sve naredbe i njihovo međusobno dejstvo.

GRAPHIC

To je funkcija koja nam služi za otvaranje različitih grafičkih modova. Njena sintaksa je sledeća:

GRAPHIC mod,cl,linija

Prvi parametar koji sledi iza komande mora biti broj od 0 do 5. U zavisnosti koji je mod u pitanju — imamo sledeće grafičke modove:

0 — tekst mod od 40 karaktera u redu,
1 — grafika visoke rezolucije (320×200),
2 — podeljen ekran na grafiku visoke rezolucije i tekst mod,
3 — višebojna grafika rezolucije (160×200),
4 — podeljen ekran na višebojnu grafiku i tekst mod,
5 — tekst mod od 80 karaktera u redu.

Od nultog do četvrtog moda sliku dobijamo na kompozitnom monitoru ili televizoru, a peti mod isključivo na RGB monitoru.

Sledeći parametar može imati samo dve vrednosti: 0 ili 1. U slučaju da je vrednost nula — dolazi samo do prelaska u izabrani mod bez brisanja slike, a ako je vrednost jedan — dolazi do brisanja sadržaja ekranske memorije. Ako ovaj parametar ne navedemo, podrazumeva se da je nula.

Poslednji parametar kod ove naredbe je linija. On se koristi isključivo kod moda 2 ili 4. Vrednost ovog parametra je u intervalu od 0 do 24. Njegova uloga je da podeli ekran na dva dela po horizon

tali, a tim što će u gornjem delu ekrana biti mod visoke rezolucije (slučaj moda 2) ili višebojna grafika (slučaj moda 4), a u donjem delu je karakter mod. U zavisnosti od ovog broja doći će do podele ekrana. To je veoma korisan parametar, jer nam omogućuje da istovremeno imamo i sliku visoke rezolucije i da možemo listati program koji je crta. Za slučaj da koristimo mod 2 ili 4, a ne navodimo parametar linije — podrazumeva se da je njegova vrednost 20.

Postoji još jedna funkcija vezana za komandu GRAPHIC. To je GRAPHIC CLR. Njena uloga je brisanje ekrana u bilo kom modu.

Primeri

GRAPHIC 1
GRAPHIC 4,1,15

Prvi primer ilustruje otvaranje grafike visoke rezolucije, a drugi višebojnu grafiku u kombinaciji sa tekstom.

COLOR

Pomoću ove naredbe možemo definisati boje sa kojima radimo. Sintaksa ove naredbe je sledeća:

COLOR osnova,boja

Kao što vidimo, naredba COLOR zahteva dva parametra i moramo ih oba postaviti. Prvi parametar mora biti broj od 0 do 6 i da ima sledeće značenje:

0 — boja pozadine,
1 — boja tačaka,
2 — boja 1 kod višebojne grafike,
3 — boja 2 kod višebojne grafike,
4 — boja bordera (okvir ekrana),
5 — boja karaktera,
6 — boja pozadine

Ako je vrednost parametra od 0 do 5, boja se odnosi na sliku koju dobijamo preko kompozitnog monitora ili televizora, a vrednosti 5 i 6 odnose se na RGB monitor.

Sledeći parametar određuje boju. Pošto COMMODORE 128 poseduje 16 različitih boja, ovaj parametar može uzimati vrednosti od 1 do 16. U zavisnosti da li koristimo kompozitni ili RGB monitor, brojevi nekih boja se razlikuju. U sledećoj tabeli daćemo brojeve i odgovarajuće boje na kompozitnom monitoru, a u zagradi boje na RGB monitoru.

1 — crna	(crna)
2 — bela	(bela)
3 — crvena	(crvena)
4 — petrolej plava	(svetla petrolej plava)
5 — ljubičasta	(svetloljubičasta)
6 — zelena	(zelena)
7 — plava	(plava)
8 — žuta	(svetložuta)
9 — narandžasta	(tamnoljubičasta)
10 — braon	(tamnožuta)
11 — svetlocrvena	(svetlocrvena)
12 — tamnosiva	(tamno petrolej plava)
13 — siva	(siva)
14 — svetlozelena	(svetloplava)
15 — svetloplava	(svetlozelena)
16 — svetlosiva	(svetlosiva)

Primeri.

COLOR 0,6:COLOR 4,8:COLOR 5,1

U ovom slučaju boja pozadine je zelena, boja bordera (okvira) je žuta, a slova su crna. Ovaj primer se odnosi na sliku koju dobijamo preko kompozitnog monitora ili televizora.

DRAW

Naredba koja nam služi za postavljanje tačke, crtanje linija i povlačenje isprekidanih linija. Sintaksa naredbe je sledeća

DRAW boja,Xkoor1,Ykoor1 TO Xkoor2,Ykoor2 ...

Prvi parametar određuje boju tačaka kojom crtamo. On može imati jednu od sledećih vrednosti:

0 — crta se bojom pozadine, tj. brišu se postojeće tačke,
1 — crta se definisanom bojom u COLOR naredbi, ili ako boja nije definisana, tačke će biti boje karaktera,
2 — u slučaju višebojne grafike boja 1,
3 — u slučaju višebojne grafike boja 2.

U slučaju da ovaj parametar ne navedemo automatski dobija vrednost 1.

Sledeća dva parametra su koordinate tačaka na ekranu. X-koordinata može biti u intervalu od 0 do 319 za mod visoke rezolucije, a u intervalu od 0 do 159 za višebojnu grafiku. U oba slučaja Y koordinata može biti u intervalu od 0 do 199. Ako navedemo samo ova dva parametra, funkcija postavlja tačku na datim koordinatama.

Sledeća dva parametra (Xkoor2 i Ykoor2) koriste se za povlačenje linije. Ograničenja za ova dva parametra su potpuno ista kao i za

Xkoor1 i Ykoor1. Dakle, ako želimo nacrtati liniju, moramo zadati koordinate njenog početka (Xkoor1, Ykoor1) i njenog kraja (Xkoor2, Ykoor2)

Ako smo koristili neke grafičke funkcije i želimo da od poslednje nacrtane tačke povučemo liniju do željenog mesta izostavićemo u naredbi prve dve koordinate i zadati samo dve poslednje (DRAW TO Xkoor2,Ykoor2).

Takođe, pored povlačenja jedne linije, ova funkcija daje mogućnost da crtamo izlomljenu liniju na sledeći način

DRAW,X1,Y1 TO X2 TO X3,Y3 TO X4,Y4

Primeri:

DRAW ,10,10 postavlja tačku na koordinatama (10,10)
DRAW ,10,20 TO 50,60 povlači liniju od (10,20) do (50,60).
DRAW ,5,10 TO 25,65 TO 120,97 povlači izlomljenu liniju, i to od (5,10) do (25,65) i od (25,65) do (120,97).
DRAW ,30,30 DRAW TO 50,80 povlači liniju od (30,30) do (50,80)

BOX

Ova grafička naredba služi za crtanje pravougaonika.

BOX boja,Xkoor1,Ykoor1,Xkoor2,Ykoor2 ugao,flag

Prvi parametar (boja) može imati sledeće vrednosti
0 — crta se bojom pozadine, tj. brišu se postojeće tačke
1 — crta se definisanom bojom u COLOR naredbi, ali ako boja nije definisana — tačke ce biti boje karaktera
2 — u slučaju višebojne grafike boja 1.
3 — u slučaju višebojne grafike boja 2

U slučaju da ovaj parametar nije dat — automatski uzima vrednost 1. Sledeća dva parametra (Xkoor1 i Ykoor1) su koordinate jednog temena, a parametri (Xkoor2 i Ykoor2) određuju suprotno teme pravougaonika. Sa ova četiri parametra pravougaonik je potpuno određen. U slučaju da ne damo koordinate (Xkoor2 i Ykoor2), naredba će podrazumevati koordinate poslednje nacrtane tačke. Za slučaj da pre ove naredbe nismo ništa crtali na ekranu podrazumeva se tačka koordinatnog početka, tj. tačka sa koordinatama (0,0)

Sledeći parametar je ugao stranice pravougaonika sa X-osom. Ugao može imati vrednost od 0 do 360 a u slučaju da nije dat — podrazumeva se da je nula.

I poslednji parametar kod ove funkcije može imati vrednosti 0 ili 1. Vrednost nula obično ne navodimo jer se podrazumeva ako nije navedena. U slučaju da je vrednost jedan, ceo pravougaonik će biti ispunjen (ofarban bojom definisanom parametrom boja)

Primeri

BOX ,10,20,50,60 — dobićemo pravougaonik kod koga gornji levi ugao ima koordinate (10,20) a donji desni ugao ima koordinate (50,60).

BOX ,10,20,50,60,45 — dobićemo isti pravougaonik kao u prethodnom primeru, samo što će biti zarotiran oko svog centra za ugao od 45 stepeni.

BOX ,30,40,60,70,,1 — u ovom slučaju pravougaonik će biti ispunjen, a pošto ugao nismo naveli — podrazumeva se da je 0

CIRCLE

Veoma složena i sa velikim mogućnostima funkcija. Pomoću nje možemo crtati kružnice, elipse, delove kružnica, delove elipsa, n-touglove upisane u kružnicu ili elipsu, kao i delove n-touglova. Elipse i n-touglovi upisani u elipsu mogu biti pod proizvodnim uglom u odnosu na X-osu.

CIRCLE boja,Xcen,Ycen,Xrad,Yrad ugao1,ugao2,rot,seg

Prvi parametar ove funkcije određuje boju kojom crtamo i može imati jednu od sledećih vrednosti:

0 — crta se bojom pozadine, tj. brišu se postojeće tačke
1 — crta se definisanom bojom u COLOR naredbi ili, ako boja nije definisana — tačke će biti boje karaktera
2 — u slučaju višebojne grafike boja 1
3 — u slučaju višebojne grafike boja 2

U slučaju da ne navedemo ovaj parametar, podrazumeva se da je njegova vrednost jedan. Sledeća dva parametra (Xcen,Ycen) određuju koordinate centra kružnice i elipse. Ako ne zadamo više nijedan parametar, onda ova funkcija postavlja tačku na zadatim koordinatama.

Sledeća dva parametra (Xrad,Yrad) određuju poluprečnik kružnice ili elipse po X-osi, tj. po Y-osi. Za crtanje kružnice dovoljno je da zadamo samo prvi parametar, dok za elipsu moramo zadati oba parametra.

Parametri ugao1 i ugao2 koriste se za crtanje delova kružnice ili elipse. Njihove vrednosti su u stepenima i nalaze se u intervalu od 0 do 360. Prvi parametar (ugao1) određuje početak crtanja, a drugi (ugao2) kraj crtanja. U slučaju da nisu zadati — podrazumeva se da je ugao1 jednak nuli, a ugao2 jednak 360.

Predzadnji parametar ove funkcije (rot) predstavlja ugao za koji se slika rotira oko svog centra. Normalno, ako crtamo punu kružnicu, ovaj parametar neće imati uticaja. Uticaj ove funkcije najbolje se vidi pri crtanju elipsi. U slučaju da ovaj parametar ne navedemo — podrazumeva se da je jednak nuli.

Poslednji parametar funkcije CIRCLE je seg. On se koristi za crtanje n-touglove upisanih u kružnicu ili elipsu. Broj stranica n-touglova

dobijamo ako podelimo 360 sa (seg), tj. seg je unutrašnji ugao stranice n-tougla. Ovde možemo koristiti jedan mali trik za crtanje kružnica ili elipsi. Ako stavimo da je seg jednako deset, dobićemo n-tougao koji ima 36 stranica. Zbog rezolucije kompjutera ovaj n-tougao će biti kružnica, ali će kompjuter ovakve kružnice nekoliko puta brže crtati nego ako definišemo običnu kružnicu. Razlika u brzini je više nego očigledna.

Zbog složenosti ove funkcije daćemo nešto više primera — da biste bolje shvatili uticaj svakog parametra.

Primeri:

CIRCLE,10,20 — dobićemo tačku na ekranu sa koordinatama (10,20).

CIRCLE,100,120,20 — ova funkcija crta kružnice kod koje su koordinate centra (100,120), a poluprečnik je 20

CIRCLE 160,100,80,10 — elipsa sa centrom (160,100). Poluprečnik po X-osi je 80, a poluprečnik po Y-osi je 10

CIRCLE,160,100,50,,90,270 — parametar Yrad nije zadat i funkcija crta kružnicu. Inače, ovaj primer crta donju polovinu kružnice jer je početni ugao 90, a krajnji 270 stepeni.

CIRCLE 160,100,80,10,,,45 — na osnovu zadatih parametara ovaj primer će nacrtati punu elipsu. Parametar rot, koji ima vrednost 45, govori nam da će elipse biti zarotirana za taj ugao.

CIRCLE,160,100,50,,,,,60 — ova funkcija će nacrtati pravilan n-tougao upisan u kružnicu sa centrom (160,100) i poluprečnikom 50. Pošto je poslednji parametar 60, a 360/60 je 6, u ovom konkretnom primeru dobićemo pravilni šestougao.

CHAR

Ova funkcija služi za ispisivanje teksta po ekranu visoke rezolucije ili višebojnom ekranu. Imamo mogućnost pozicioniranja teksta i flag koji određuje da li je tekst normalan ili inverzan

CHAR boja,Xkar,Ykar,text$,inv

Prvi parametar je isti kao i kod prethodnih funkcija i može imati sledeće vrednosti

0 — crta se bojom pozadine, tj. brišu se postojeće tačke
1 — crta se definisanom bojom u COLOR naredbi ili, ako boja nije definisana — tačke će biti boje karaktera.
2 — u slučaju višebojne grafike boja 1
3 — u slučaju višebojne grafike boja 2

Ako ovaj parametar ne navedemo, podrazumeva se da je njegova vrednost jedan.

Sledeća dva parametra Xkar i Ykar određuju poziciju prvog karaktera u tekstu. Ovde je važno napomenuti da vrednost Xkar mora biti u intervalu od 0 do 39 (do 79 ako koristimo RGB monitor), a vred-

nost Ykar od 0 do 24. To znači da nemamo mogućnost pomeranja teksta za jednu tačku, već samo po karakterima.

Tekst koji želimo ispisati ima oznaku text$. Znak za dolar ($) znači da se tekst mora nalaziti pod znacima navoda ili kao alfanumerički podatak u nekoj promenljivoj.

I poslednji parametar može imati dve vrednosti: nula ili jedan. U slučaju da je nula — tekst će biti ispisan normalnim slovima, a u slučaju da je jedan — tekst će biti napisan inverznim slovima. Ako pak nije zadat — podrazumeva se da je ovaj parametar jednak nuli.

Primeri:

CHAR,10,5,"COMMODORE" ispisaće COMMODORE

CHAR,14,6,"128",1 — ako ove dve funkcije napišemo zajedno, onda će ispod natpisa COMMODORE biti broj 128, ali zbog parametra inv koji je jedan, broj će biti napisan inverzno.

Napomenimo vam još da ova funkcija može da se koristi pored grafičkog moda i u tekst modu. Pri pravljenju neke tabele možemo je iskoristiti za pozicioniranje slova na ekranu.

PAINT

Funkcija koja ima ulogu da određeni deo ekrana oboji zadatom bojom. Deo ekrana mora biti potpuno uokviren.

PAINT boja,Xkoor,Ykoor,mod

Krenimo redom — da vidimo koja je uloga parametara kod ove funkcije. Prvi parametar određuje koju boju koristimo i on može imati jednu od sledećih vrednosti:

0 — crta se bojom pozadine, tj. brišu se postojeće tačke
1 — crta se definisanom bojom u COLOR naredbi ili ako boja nije definisana — tačke će biti boje karaktera.
2 — u slučaju višebojne grafike boja 1
3 — u slučaju višebojne grafike boja 2.

Sledeća dva parametra Xkoor i Ykoor određuju tačku na ekranu od koje počinje bojenje. Veoma je važno da tačno zadamo koordinate ove tačke. Da bismo uspešno obojili neku površinu, ona mora biti sa svih strana zatvorena, a tačka koju zadajemo PAINT funkcijom mora biti unutar uokvirenog dela ekrana.

Poslednji parametar (mod) ima ulogu samo kod višebojne grafike. U modu visoke rezolucije on nema nikakav uticaj na bojenje. Mod može imati samo dve vrednosti (nula ili jedan).

0 — obojiće deo ekrana koji je uokviren bojom koja je zadata PAINT funkcijom. Linije nacrtane drugom bojom funkcija neće uzimati u obzir. Pri korišćenju ovog moda moramo strogo voditi računa da polje koje bojimo bude sa svih strana uokvireno određenom bojom.

1 — u ovom modu funkcija boji deo ekrana koji je uokviren bilo kojom bojom. Površ čak ne mora biti uokvirena istom bojom. Kao i u modu nula, boja se bojom koja je definisana PAINT funkcijom.

Primeri

GRAPHIC 1,1 CIRCLE,100,100,50 PAINT,100,100

Kada izvršimo ove funkcije kružnica sa centrom u (100,100) i poluprečnikom 50 biće obojena bojom karaktera (jer boja nije definisana).

LOCATE

Ova funkcija ima ulogu da postavi koordinate kursora tačke. Ona sa datim koordinatama ne postavlja tačku, ali kompjuter te koordinate registruje kao poslednje na kojima je postavljena tačka.

LOCATE Xkoor,Ykoor

Ovu funkciju mnogo je lakše objasniti preko primera.

GRAPHIC 1,1 LOCATE 100,100 DRAW TO 10,30

Ovaj skup funkcija povlači liniju od koordinate (100,100) do koordinate (10,30).

Inače, ova funkcija ima drugu ulogu: predviđena je za postavljanje fiksne koordinate, a dalje crtanje vršilo bi se relativnim koordinatama.

SSHAPE

Ova funkcija služi za prebacivanje dela ekrana visoke rezolucije u alfanumeričku promenljivu.

SSHAPE promenljiva$,Xkoor1,Ykoor1,Xkoor2,Ykoor2

Veoma korisna grafička funkcija koju možemo koristiti kako u radu sa grafikom, tako i pri kreiranju sprajtova. Pošto se u ovom poglavlju bavimo isključivo grafikom, predstavićemo samo njene grafičke mogućnosti. U poglavlju o sprajtovima daćemo njenu primenu u radu sa sprajtovima. Krenimo redom: prvi parametar je bilo koja alfanumerička promenljiva. Znak za dolar iza promenljive je obavezan. Iza tog parametra slede koordinate dve tačke na ekranu (Xkoor1,Ykoor1) i (Xkoor2,Ykoor2). Ova funkcija ima zadatak da deo ekrana visoke rezolucije prebaci u alfanumeričku promenljivu. Deo ekrana koji se na ovaj način memoriše je pravougaonik, čije gornje levo teme ima koordinate (Xkoor1,Ykoor1), a donje desno teme ima koordinate (Xkoor2,Ykoor2).

Pri korišćenju ove funkcije veoma je važno da broj bajta koji je ograničen pravougaonikom ne bude veći od 255, jer je to maksimalni broj bajta koji može da sadrži alfanumerička promenljiva. Ako, pak, želimo da memorišemo veći deo ekrana, možemo da koristimo istu funkciju, ali moramo koristiti drugu promenljivu. Na ovaj način možemo memorisati ceo ekran visoke rezolucije. Prethodno ga moramo podeliti na pravougaonike, a zatim svaki pravougaonik dodeliti drugoj promenljivoj. Naravno, za memorisanje možemo koristiti i niz alfanumeričkih promenljivih (na primer A$(I)). Primer:

SSHAPE a$,10,15,20,30

Ova funkcija sama za sebe nema nikakvog smisla, jer šta da radimo sa dêlom ekrana koji smo memorisali. Zbog toga sa njom u pâru ide i sledeća funkcija.

GSHAPE

Ova funkcija se koristi u pâru sa prethodnom funkcijom SSHAPE. Svaka od njih, sama za sebe nema nikakvog smisla, a kada se koriste zajedno — možemo postići veoma korisne grafičke efekte. Njena uloga je da deo ekrana koji je memorisan funkcijom SSHAPE u alfanumeričku promenljivu vrati na ekran visoke rezolucije na željeno mesto.

GSHAPE promenljiva$ Xkoor,Ykoor,mod

Ova funkcija je na neki način suprotna prethodnoj. Parametar promenljiva$ mora biti onaj isti koji smo koristili kod funkcije SSHAPE. Nakon toga zadajemo koordinate tačke parametrima (Xkoor, Ykoor). To su tačke gornjeg levog ugla pravougaonika koji je memorisan funkcijom SSHAPE. Koordinatama te tačke određujemo mesto na ekranu gde želimo da postavimo prethodno memorisani deo ekrana. Na ovaj način jedan deo ekrana možemo iskoristiti na bilo kom mestu, i to neograničeni broj puta.

Kopiranje možemo izvršiti na 5 različitih načina korišćenjem parametra mod. On može imati sledeće vrednosti:

0 — novodobijena slika će biti potpuna kopija originalnog dêla memorije koji je memorisan.

1 — dobijena slika će biti inverzna u odnosu na originalni deo ekrana koji je memorisan.

Za ove dve vrednosti parametra mod, kompletan pravougaonik se preslikava na željeni deo memorije i pri tom se brišu sve tačke koje su se prethodno nalazile u tom dêlu memorije. Tako, na primer, ako smo memorisali prazan ekran i izvršimo preslikavanje — doći će do brisanja dêla ekranske memorije gde smo prebacili pravougaonik. Znači, kada parametar mod ima vrednost 0 ili 1, na preslikanu memoriju uopšte ne utiče prethodno stanje ekrana tog dêla memorije. Kod sledeće tri vrednosti parametra mod prethodno stanje ekrana ima uticaj na preslikani deo memorije.

2 — kada je ovo vrednost parametra mod, pravougaonik gde se vrši preslikavanje dobiće sve tačke izvornog pravougaonika, a takođe će i zadržati sve tačke koje su se nalazile u tom dêlu ekranske memorije pre izvršavanja ove funkcije. Jednom rečju — funkcija vrši matematičko OR memorisanog dêla ekrana i dêla ekrana gde se vrši preslikavanje

3 — u ovom slučaju pri preslikavanju se koriste matematička funkcija AND. To znači da će se u novodobijenom dêlu ekrana pojaviti samo tačke koje se nalaze i u izvornom pravougaoniku i u dêlu memorije gde se vrši preslikavanje

4 — i na kraju, pri ovoj vrednosti parametra mod primeniće se matematička funkcija EOR (ekskluzivno ili). To znači da će se pojaviti samo tačke koje se nalaze ili na originalnom pravougaoniku ili u dêlu ekranske memorije na koji se vrši preslikavanje. U slučaju da se tačka nalazi na oba mesta — biće obrisana.

Korišćenjem ove dve funkcije možemo veoma lako i brzo dobiti ekran visoke rezolucije kod koga su pojedini delovi isti ili veoma slični. Da bi ste lakše razumeli ove dve funkcije dajemo kraći primer:

```
GRAPHIC1 i CIRCLE 15,15,10 SSHAPE A$ 5,5,25,25
GSHAPE A$,100,100,0
```

Ovaj primer otvara ekran visoke rezolucije, crta kružnicu koja se zatim memoriše u promenljivu A$ i na kraju preslikava u drugi deo ekranske memorije

SCNCLR

Funkcija koja ima zadatak da obriše ekransku memoriju.

SCNCLR mod

Ova funkcija je na neki način dupla, jer funkcija GRAPHIC ima mogućnost da pored ulaska u određeni grafički mod ujedno briše ekran kod tog moda. Jedina razlika je u tome što primenom ove funkcije možemo biti u jednom grafičkom modu a brisati ekransku memoriju drugog grafičkog moda ne napuštajući prvi mod. Tako, na primer, ako radimo sa RGB monitorom i imamo 80-kolonski ekran, može se izvršiti brisanje ekrana visoke rezolucije bez ulaženja u taj mod. Koji ekran brišemo — određuje parametar mod. On može imati jednu od sledećih 6 vrednosti:

0 — tekst-mod od 40 karaktera u redu
1 — grafika visoke rezolucije (320×200)
2 — podeljen ekran na grafiku visoke rezolucije i tekst-mod
3 — višebojna grafika rezolucije (160×200)
4 — podeljen ekran na višebojnu grafiku i tekst-mod
5 — tekst-mod od 80 karaktera u redu

Kod ove funkcije nije neophodno da zadamo parametar mod. Za slučaj da ga nismo zadali, već samo otkucamo SCNCLR — funkcija će obrisati ekransku memoriju koju upravo koristimo

WIDTH

Ova naredba kontroliše debljinu linija kojom crtamo u grafici visoke rezolucije.

WIDTH n

Ova funkcija poseduje samo jedan parametar, koji može imati dve vrednosti (1 ili 2). Kada parametar 'n' ima vrednost jedan, crtanja se izvršava potpuno normalno, kao da ova funkcija ne postoji. Prava uloga funkcija sa parametrom jedan je da isključuje mod u koji je grafika dovedena korišćenjem ove funkcije sa parametrom dva.

Nakon startovanja ove funkcije sa parametrom dva, sve tačke koje budemo crtali biće duplo šire od normalnih. Zadebljanje linija vrši se samo po X-osi, dok po Y-osi one ostaju nepromenjene. Da biste lakše shvatili dejstvo ove funkcije, otkucajte sledeći primer i na ekranu ćete videti manifestaciju ove funkcije.

```
10 GRAPHIC 1,1
20 CIRCLE ,160,100,50
30 WIDTH 2
40 CIRCLE ,170 100 50
```

Nakon startovanja ovog programa na ekranu ćemo dobiti dve kružnice. Desna kružnica je duplo deblja od leve. Da bismo grafički mod vratili u normalno stanje, dovoljno je da otkucamo WIDTH 1.

SCALE

Ova funkcija ima zadatak da menja koordinatni sistem kod ekrana visoke rezolucije. Na primer, ako nekim matematičkim proračunom dobijemo tačku koja ima koordinate (700,300) i pokušamo tu tačku da nacrtamo, kompjuter će nam javiti poruku o grešci. Pomoću ove funkcije možemo na ekranu visoke rezolucija predstaviti ovu tačku. Odmah da napomenemo, da ne bi došlo do zabune, funkcija SCALE ne menja stvarnu rezoluciju ekrana (320×200 za visoku rezoluciju ili 160×200 za višebojnu grafiku), već samo menja razmeru koordinatnog sistema.

SCALE flag,Xmax,Ymax

Parametar flag koji sledi odmah iza naredbe može imati samo dve vrednosti (0 ili 1). On služi za uključivanje ove funkcija (flag=1), odnosno za isključivanje (flag=0, povratak na normalni koordinatni sistem). Sledeća dva parametra predstavljaju maksimalne vrednosti koje se mogu pojaviti na X odnosno Y-osi.

Kod grafike visoke rezolucije širina ekrana je podeljena na 320 tačaka, i to od 0 do 319. Ako otkucamo sledeće

SCALE 1,640 200

širina ekrana ima takođe 320 tačaka (rezolucija se ne menja), ali je izvršena podela od 0 do 640 sa korakom dva (0,2,4,6, ...). Podela po Y-osi je od 0 do 200, što znači da je ostala nepromenjena. Na ovaj način možemo predstaviti razne matematičke funkcije čiji su parametri veći od 320 po X-osi i veći od 200 po Y-osi.

Maksimalne vrednosti parametra Xmax ili Ymax kod ove funkcije su 32767. Pošto ova funkcija nema mogućnost umanjivanja ekrana, minimalne vrednosti za parametar Xmax i Ymax su redom 320 i 200.

Mogućnost koju nam pruža ovaj program možemo postići i na drugi način. Koordinate tačaka koje dobijamo možemo pre crtanja deliti nekom konstantom. Međutim, to bi dosta usporilo program, a bilo koja izmena zahtevala bi veću promenu.

Ako brojevi Xmax i Ymax nisu zadati, kompjuter automatski podrazumeva da je Xmax=1023, a Ymax=511. Normalno — da bi ovo funkcionisalo parametar flag mora biti jednak jedinici.

RCLR

Funkcija koja nam daje informaciju o bojama koje koristimo.

RCLR (broj)

Iza nje u zagradi stoji parametar broj koji može imati sedam vrednosti. U zavisnosti koju vrednost ima, ova funkcija daje informaciju o boji željenog dela ekrana ili grafičkog moda. Iz sledeće tabele možemo videti koju nam informaciju pruža funkcija RCLR u zavisnosti od parametra *broj*.

Broj	Informacija
(0)	Boja ekrana u tekst-modu (40 karaktera)
(1)	Boja tačaka u modu visoke rezolucije
(2)	Boja 1 kod višebojne grafike
(3)	Boja 2 kod višebojne grafike
(4)	Boja okvira (bordera) kod ekrana od 40 karaktera
(5)	Boja karaktera (40 ili 80-karakterni tekst)
(6)	Boja ekrana u tekst modu (80 karaktera)

Na primer, ako otkucamo PRINT RCLR(6), dobićemo broj 1, što znači da je boja ekrana kod tekst moda od 80 karaktera crna (jedinica je oznaka za crnu boju). Inače ova je funkcija suprotna funkciji COLOR.

RDOT

Ova je funkcija suprotna funkciji LOCATE. Dok naredba LOCATE postavlja poziciju grafičkog kursora, ova funkcija ima zadatak da čita tu vrednost.

RDOT (broj)

Iza naredbe u zagradi sledi parametar broj koji može imati tri vrednosti:

(0) Funkcija nam daje podatak o položaju grafičkog kursora po X-osi.

(1) Funkcija nam daje podatak o položaju grafičkog kursora po Y-osi.

(2) Funkcija nam daje podatke o boji tačke koja se nalazi na položaju grafičkog kursora.

Pri korišćenju relativnih koordinata veoma je bitno da znamo tačnu poziciju grafičkog kursora. Na sledećem primeru se najbolje vidi dejstvo ove funkcije

```
DRAW,10 10 TO 50,70 PRINT RDOT(0) RDOT(1)
```

Na ekranu ćemo dobiti sledeće vrednosti 50 i 70, što znači da grafički kursor tj poslednja nacrtana tačka ima sledeće koordinate (50,70).

RGR

Ova je funkcija suprotna funkciji GRAPHIC. Njen zadatak je da nam da informaciju u kom grafičkom modu se nalazimo

RGR (n)

Iza funkcije, u zagradi, sledi parametar čija vrednost nije bitna za značenje funkcije. Firma *Commodore* je verovatno uvela ovaj parametar radi kasnijih proširenja. Slobodno možemo pisati da je 'n' jednako 0. Kao rezultat ove funkcije dobijamo brojeve od 0 do 5 i iz sledeće tabele možemo videti u kom se grafičkom modu nalazimo

0 — tekst mod od 40 karaktera u redu
1 — grafika visoke rezolucije (320×200)
2 — podeljen ekran na grafiku visoke rezolucije i tekst mod
3 — višebojna grafika rezolucije (160×200)
4 — podeljen ekran na višebojnu grafiku i tekst mod.
5 — tekst mod od 80 karaktera u redu

Relativne koordinate

Na kraju poglavlja o grafici objasnićemo i rad sa relativnim koordinatama. Kod svake grafičke funkcije gde postavljamo koordinate neke tačke — možemo koristiti pored apsolutnih i relativne koordinate. Ako ispred broja za koordinatu stavimo predznak + ili —, te koordinate postaju relativne. Ovo je najbolje objasniti na konkretnom primeru

```
DRAW,10,20 TO +50,+30
```

ova funkcija će povući duž od tačke sa koordinatama (10,20) do tačke (60 50), jer je 60=10+50 a 50=20+30

Kao što vidite, kada postoji predznak + ili —, nove koordinate dobijamo tako što brojeve iza predznaka dodajemo ili oduzimamo koordinati poslednje nacrtane tačke. Ali, pri radu sa relativnim koordinatama postoji jedna greška. Kada koristimo relativne koordinate sa predznakom +, sve funkcioniše bez greške, ali ako koristimo predznak — kompjuter će nam javiti grešku ILLEGAL QUANTITY. Firma COMMODORE obećava da će ovu grešku otkloniti u što je moguće kraćem roku.

5.12. SPRAJTOVI

U ovom poglavlju objasnićemo rad sa sprajtovima. Sprajt predstavlja malu sliku (veličine 24 tačke po horizontali i 21 tačku po vertikali) koja se može nalaziti bilo gde na ekranu. On je potpuno nezavisan od grafičkog moda u kome se kompjuter nalazi. To znači da sprajtove možemo dobiti u karakter-modu, modu visoke rezolucije i u višebojnom modu. Mogućnost da na ekranu imamo sprajtove ostvaruje VIC II čip. Pod njegovom kontrolom je slika na televizoru i kompozitnom monitoru. Sliku na RGB monitoru kontroliše drugi čip koji nema mogućnost prikazivanja sprajtova. To znači da u modu od 80 karaktera na ekranu ili modu visoke rezolucije (640×200) nemamo mogućnost rada sa sprajtovima.

Commodore 128 ima mogućnost da na ekranu prikaže najviše osam sprajtova. Oni su međusobno potpuno nezavisni. To znači da svaki možemo različito definisati i oni se mogu nalaziti bilo gde na ekranu — potpuno nezavisno jedan od drugog. Napomenućemo da je korišćenjem mašinskog jezika moguće postići više od osam sprajtova na ekranu. Ali tada se koristi jedan trik. Jedan sprajt velikom brzinom menja svoje mesto i oblik. Zato imamo utisak da na ekranu postoji više od jednog sprajta, iako je korišćen samo jedan sprajt. Na taj način možemo postići 16 ili 24 sprajta. Ali tema ovog poglavlja je rad sa sprajtovima u Basic-u, tako da ćemo raditi sa najviše osam sprajtova.

Pored toga što tu malu sličicu nazvanu *sprajt* možemo postaviti bilo gde na ekranu, možemo joj i dodeliti prioritet. Commodore-ov ekran sastoji se iz tri nivoa: prednji plan, središnji deo i pozadina. Ako na ekranu postoji neki tekst ili crtež, mi na taj ekran možemo postaviti sprajt na dva različita načina. Prvi način: sprajt se nalazi ispred ekrana tako da zaklanja deo ekrana koji se nalazi iza njega. To znači da sprajt ima prioritet u odnosu na ekran. Drugi način: sprajt se nalazi iza ekrana, a to znači da delovi ekrana zaklanjaju sprajt. Na ovaj način postiže se odličan utisak trodimenzijalnosti. Takođe, kao što sprajt može imati prioritet u odnosu na ekran, ili ekran da ima prioritet u odnosu na sprajt — takav odnos postoji i između dva sprajta. Kada se dva sprajta kreću jedan ka drugome, doći će do njihovog preklapanja. Koji će sprajt biti gore a koji dole — veoma lako možemo definisati. Kod video-igara se najbolje vidi međusobni uticaj sprajtova i pozadine, kao i međusobni uticaj dva sprajta.

Pošto smo sprajt definisali kao mali ekran, a poznato nam je da ekran može biti visoke rezolucije ili da je višebojni, nameće nam se pitanje — šta na raspolaganju imamo kod sprajtova. I kod njih, kao i kod ekrana, postoje dva moda. To su normalni mod, kod koga je sprajt veličine 24 tačke po horizontali i 21 tačka po vertikali, i višebojni mod gde je veličina sprajta fizički ostala ista ali se broj tačaka po horizontali prepolovio tako da je rezolucija kod višebojnog sprajta 12 tačaka po horizontali i 21 tačka po vertikali. U prvom modu, isto kao i kod ekrana visoke rezolucije, imamo na raspolaganju samo dve boje, dok kod drugog moda imamo mogućnost korišćenja četiri različite boje. U prvom modu jedna tačka na ekranu odgovara jednom bitu u memoriji gde je sprajt definisan, dok u višebojnom modu jednoj tački na ekranu odgovaraju dva bita u memoriji gde je sprajt definisan. Sa dva bita možemo postići četiri različite kombinacije: 00, 01, 10 i 11. Sada je jasno zašto kod višebojnog moda imamo četiri različite boje. U oba slučaja memorija gde je sprajt definisan — veličine je 64 bajta (24×21/8=64 ili 12×21/8 delimo sa osam, jer jedan bajt ima osam bita). Pored boje, sprajtovi se razlikuju i po veličini. Imamo mogućnost da sprajt duplo uvećamo po X ili Y-osi. Takođe ga možemo uvećati i po X i po Y-osi. Na taj način dobićemo sprajt koji je jednak veličini četiri sprajta. I on zauzima 64 bajta memorije jer ima isti broj tačaka kao i normalni sprajt, samo što je kod njega svaka tačka udvojena kako po horizontali, tako i po vertikali.

Pre nego što pređemo na objašnjavanje pojedinih funkcija, napominjemo još jednom da su sprajtovi međusobno nezavisni, što znači da pored položaja, prioriteta, kretanja i veličine — neki mogu biti jednobojni, a neki višebojni (po želji).

SPRDEF

Naredba koja služi za definisanje oblika sprajta

SPRDEF

Karakteristično za ovu naredbu je da nema nijedan parametar i da ne spada u klasične BASIC funkcije. Izvršenjem ove naredbe kompjuter prelazi u poseban mod tzv. sprajt-editor. Kada je startujemo, na ekranu će se pojaviti pravougaonik veličine 24×21 karakter, i kompjuter će nas pitati koji sprajt želimo da definišemo. Tada možemo izabrati jedan od osam sprajtova pritiskanjem broja od 1 do 8. Važno je napomenuti da sprajt-editor daje sliku isključivo na televizoru ili kompozitnom monitoru. Na RGB monitoru ne možemo raditi sa sprajt-editorom, jer čip koji kontroliše njegovu sliku ne može da podržava rad sa sprajtovima.

Kada smo izabrali željeni sprajt, on će se na ekranu pojaviti u gornjem desnom uglu pored velikog pravougaonika (24×21 karakter). Sada ćemo navesti skup komandi koje se nalaze u sklopu sprajt-editora.

Kursori: U velikom pravougaoniku nalazi se znak za plus (+) koji kursorima pomeramo po celom pravougaoniku. Dovodimo ga na željeno mesto i tu možemo postaviti ili obrisati tačku.

RETURN Taster (RETURN) pomera kursor na početak sledeće linije

HOME Ovaj taster pomera kursor u gornji levi ugao pravougaonika. Ima istu funkciju kao u normalnom tekst-modu.

CLR. Ova funkcija služi za brisanje celog sprajta i dovođenje kursora u gornji levi ugao pravougaonika. I ona ima istu funkciju kao i u karakter-modu.

Boja Ispod tastera od 1 do 8 može se primetiti da su ispisane boje u dva reda. Kao i u karakter-modu, pomoću ovih tastera možemo birati boju. Istovremenim pritiskom na tastere CTR i brojeve od 1 do 8 dobijamo boje iz gornjeg reda, a pritiskom na Commodore-ov taster i brojeve od 1 do 8 dobijamo boje iz donjeg reda.

1 do 4 Tasteri jedan do četiri služe za definisanje sprajta, i to:
(1) briše tačku na sprajtu,
(2) postavlja tačku na sprajtu,
(3) postavlja tačku boje jedan kod višebojnog sprajta,
(4) postavlja tačku boje dva kod višebojnog sprajta

A ova funkcija uključuje odnosno isključuje automatsko ponavljanje pritisnutog tastera za postavljanje (odnosno brisanje) tačaka na sprajtu (1—4). Ako je automatsko ponavljanje uključeno pritiskom, na primer na taster dva, kompjuter će automatski početi da postavlja tačke jednu za drugom. U slučaju da je automatsko postavljanje isključeno, kompjuter će postaviti jednu tačku, a za sledeću moraćemo da pustimo taster i da ga ponovo pritisnemo

C Ova funkcija služi za kopiranje definicije jednog sprajta na drugi. Tako ako želimo da imamo više istih sprajtova dovoljno je da definišemo samo jedan, a uz pomoć ove funkcije definicija sprajta možemo veoma lako preneti na druge sprajtove.

M Upoznali smo vas već da postoje dve vrste sprajtova. obični i višebojni. Pritiskom na taster M, sprajt koji definišemo prelazi iz jednog moda u drugi. Tako, dok definišemo sprajt možemo odrediti da li će on u našem programu biti jednobojni ili višebojni

X i Y Ova dva tastera služe da sprajt uvećaju ili umanje po X odnosno Y-osi. Ako je sprajt normalne veličine i ako pritisnemo taster X, automatski će se svaka tačka po X-osi udvostručiti i sprajt će biti duplo duži. Potpuno analogno važi i za Y-osu. Napominjemo da sprajt može biti normalne veličine uvećan po X-osi, uvećan po Y-osi ili uvećan i po X i po Y-osi

STOP Taster STOP nam služi da promenimo sprajt koji definišemo. Kada ga pritisnemo dobićemo isto pitanje kao i na početku. koji sprajt želimo da definišemo

SHIFT RETURN Istovremenim pritiskom na tastere SHIFT i RETURN napuštamo sprajt-editor. Ovim pritiskom mi ga još uvek nismo napustili, ali ako nakon toga pritisnemo RETURN, ponovo smo došli u karakter-mod, gde možemo nastaviti sa pisanjem programa ili upotrebiti jednu od narednih funkcija da bismo ovako definisani sprajt snimili

SPRITE

Ova funkcija služi prvenstveno za uključivanje pojedinog sprajta

SPRITE broj,on/off,boja,prioritet,X-pov,Y-pov,više-boja

Da bismo sprajt videli na ekranu, on mora biti uključen. Prvi parametar (broj) određuje broj sprajta sa kojim želimo da radimo. On može biti od 1 do 8.

Sledeći parametar uključuje ili isključuje sprajt. Ako je njegova vrednost 1, sprajt je uključen i vidi se na ekranu, a ako je pak 0 — sprajt je isključen. Ova dva parametra su neophodna i njih moramo navesti a ako nam ostali nisu potrebni — ne moramo ih ni navesti.

Parametar *boja* određuje koje će boje biti sprajt. On može imati vrednost od 1 do 16, a određuje boju tačke kod običnog sprajta ili osnovnu boju kod višebojnog sprajta, tj. boju tačaka koja je određena sa dva bita koji su (oba) jednaki jedinici.

Sledeći parametar određuje prioritet sprajta. Ako je jednak nuli, sprajt će biti ispred ekrana, tj. u prednjem planu a ako je jednak jedinici sprajt će se nalaziti iza ekrana.

Parametri X-pov i Y-pov određuju povećanje sprajta po X odnosno Y-osi. Kod normalnog sprajta oba parametra su jednaka nuli. Mogu imati vrednost 0 ili 1. U slučaju da su oba jednaki jedinici, sprajt će biti uvećan.

I poslednji parametar određuje da li je sprajt običan ili višebojan. Ako nije naveden, podrazumeva se da je jednak nuli a to znači da je sprajt običan. U slučaju da je jednak jedinici, reč je o višebojnom sprajtu.

Uz pomoć ove funkcija sprajt je postavljen na ekranu određene veličine i prioriteta. Sledeće funkcije određuju njegovu boju (ako je višebojan) i kontrolišu njegovo kretanje.

SPRCOLOR

Jednobojni sprajtovi imaju potpuno nezavisnu boju koju određuje gornja funkcija. Ova funkcija služi za određivanje boje isključivo kod višebojnih sprajtova.

SPRCOLOR boja1,boja2

Iza naredbe slede dva parametra *boja1* i *boja2*. Pošto postoji samo jedna funkcija za boju a osam sprajtova, to znači da višebojni sprajtovi moraju imati po dve boje iste. Ovom funkcijom određujemo te dve boje. Oba parametra se nalaze u intervalu od 1 do 16 i oni određuju boje koje su tabelarno date u odeljku o grafici. Treća boja (osnovna boja sprajta) određuje se funkcijom SPRITE i potpuno je nezavisna u odnosu na boje ostalih sprajtova. Poslednja, četvrta boja je boja pozadine, a to znači — boja ekrana. Ona zavisi od položaja sprajta. To su providna mesta na sprajtu.

SPRSAV

Ovu funkciju koristimo za definisanje sprajta, da definiciju sprajta prebacimo u alfanumeričku promenljivu ili da definiciju jednog sprajta prebacimo na drugi sprajt.

```
SPRSAV broj,promenljiva$
SPRSAV promenljiva$,broj
SPRSAV broj1,broj2
```

Kao što vidite, ova funkcija ima trostruku ulogu. Krenimo redom. Prva funkcija ima zadatak da definiciju sprajta prebaci u neku alfanumeričku promenljivu. Prvi parametar određuje broj sprajta, a drugi, u koju promenljivu želimo da prebacimo njegovu definiciju. Primenom ove funkcije može se više definicija sprajtova, koji predstavljaju određenu animaciju, prebaciti u niz alfanumeričkih promenljivih. Ovaj niz možemo snimiti na disk ili kasetofon kao sekvencijalni zapis, i na taj način sačuvati definicije sprajtova.

Druga funkcija ima obrnutu ulogu. Njen zadatak je da vrednost alfanumeričke promenljive prebaci u sprajt. Sa ovom funkcijom je povezana grafička naredba SSHAPE, koja prebacuje deo ekrana visoke rezolucije u alfanumeričku promenljivu. Kombinacijom ova dve funkcije možemo deo ekrana visoke rezolucije prebaciti u sprajt. Prvi parametar određuje alfanumeričku promenljivu, a drugi parametar određuje broj sprajta.

I treća funkcija ima zadatak da definiciju jednog sprajta iskopira na neki drugi sprajt. Koristimo je kad želimo da imamo više istih sprajtova. Definišemo jedan, a onda primenom ova funkcije definišemo i ostale sprajtove. Ona poseduje dva parametra koji mogu imati vrednosti od 1 do 8. Njenom primenom, definicija sprajta koji je određen parametrom *broj1* preslikava se u sprajt koji je određen parametrom *broj2*.

RSPRITE

Ovo je suprotna funkcija funkciji SPRITE. Ona nam daje informaciju o sprajtu.

```
RSPRITE (broj,parametar)
```

Iza funkcije u zagradi moramo navesti dva parametra. Prvi parametar ima ulogu da odredi broj sprajta čiji nas parametri zanimaju. Drugi parametar može imati vrednosti od 0 do 5, u zavisnosti od toga koji nas parametar kod određenog sprajta zanima. Od vrednosti drugog parametra dobijamo sledeće informacije:

0 — Da li je sprajt uključen ili isključen (0 ili 1)
1 — Osnovna boja sprajta (1 do 16).
2 — Prioritet sprajta u odnosu na pozadinu (0 ili 1).

3 — Uvećanje sprajta po X-osi (0 ili 1)
4 — Uvećanje sprajta po Y-osi (0 ili 1)
5 — Običan ili višebojni sprajt (0 ili 1).

Najbolje je vrednost ove funkcije dodeliti nekoj numeričkoj promenljivoj — na primer

V=RSPRITE (1,5)

Dalje se promenljiva V koristi u programu i, u zavisnosti od njene vrednosti, program se grana i dobijamo željene efekte

RSPCOLOR

Suprotna funkcija funkciji SPRCOLOR. Njen zadatak je da nam da informaciju o bojama višebojnog sprajta

RSPCOLOR (broj)

Iza funkcije, u zagradi, sledi parametar *broj*. On može imati dve vrednosti (1 ili 2) u zavisnosti od toga koja nas boja kod višebojnog sprajta interesuje. Ako je, na primer, vrednost parametra b r o j 1, kao rezultat dobićemo broj prve boje kod višebojnog sprajta. Kao i kod prethodne funkcije, najbolje je njenu vrednost dodeliti nekoj numeričkoj promenljivoj — na primer

B=RSPCOLOR (1)

Možemo je veoma korisno iskoristiti kada se sprajt kreće i nalazi na deo ekrana koji je iste boje kao i sam sprajt. Kad to ustanovimo čitanjem promenljive B, boju sprajta možemo promeniti kako bi on ostao vidljiv

MOVSPR

Ova naredba nam služi za postavljanje ili pomeranje sprajta po ekranu

```
MOVSPR broj Xkoor Ykoor
MOVSPR broj +/—Xkoor +/—Ykoor
MOVSPR broj dužina ugao
MOVSPR broj ugao #brzina
```

Kao što vidimo, ona se može primeniti na četiri različita načina. Prvi parametar iza naredbe predstavlja broj sprajta koji želimo da koristimo. Taj parametar je zajednički za sva četiri načina postavljanja ili kretanja sprajta. Sada krenimo redom da vidimo sva četiri načina kretanja sprajtova.

(1) Iza naredbe sledi broj sprajta koji želimo da pomerimo. Nakon toga sledi koordinata tačke gde želimo da bude naš sprajt. X-koordinata, kod ove naredbe, mora biti u intervalu od 0 do 511, dok Y-koor-

dinata mora biti u intervalu od 0 do 255. Preciznost pomeranja sprajta je jedna tačka. Možda vam izgleda da je preciznost pomeranja sprajta veća, jer sprajt po X koordinati može imati 512 položaja. Međutim — nije tako. Sprajt ima mogućnost kretanja van opsega ekrana. Ako bismo sprajt kretali po X koordinati od 0 do 512, on se u početku ne bi video. Nakon toga bi se na levoj strani ekrana počeo pojavljivati, da bi na kraju počeo nestajati na desnoj strani. Vizuelni deo kretanja sprajta je u intervalu od 24 do 344 po X-koordinati, i od 50 do 250 po Y koordinati. U svim ostalim slučajevima on se ne vidi jer je iza ekrana. Napomenućemo da u mašinskom jeziku možemo postići da se sprajt vidi iako je iza ekrana.

(2) Naredba veoma slična prethodnoj. Razlika je u tome što kod prethodne naredbe zadajemo apsolutne koordinate sprajta, a kod ove relativne. Predznaci + ili — ispred vrednosti koordinate znače da radimo u relativnim koordinatama. Rad sa ovakvim koordinatama je isti kao i kod grafičkih funkcija, samo što ovde nema greške. Ako na primer, imamo funkciju: MOVSPR 1 +10,—15, sprajt će zauzeti nove koordinate koje su za 10 uvećane po X-osi, a za 15 umanjene po Y-osi u odnosu na prethodnu poziciju sprajta.

(3) I ova naredba kao i prethodna radi sa relativnim koordinatama. To znači da će pozicija sprajta zavisiti od njenog prethodnog mesta na ekranu. Ovde takođe zadajemo koordinate tačke ali ne koristimo X i Y-osu već dužinu i ugao. Tako ovu funkciju primenjujemo kada želimo da neki sprajt pomerimo u odnosu na njegov trenutni položaj za određeno rastojanje i u željenom pravcu. Na primer, ako želimo da sprajt pomerimo za deset tačaka gore desno, napisaćemo da je dužina 10, a ugao 45 stepeni. Vodite računa da se kod ove naredbe parametri *dužina* i *ugao* ne odvajaju zarezom već tačka-zarezom (;). Na taj način se razlikuje od prve naredbe u sintaksi.

(4) Poslednji način predstavlja kretanje sprajta. Iza broja sprajta slede dva parametra ugao i brzina kretanja (važno je da ispred broja za brzinu kretanja stoji znak numerik #). Parametar ugao mora biti u intervalu od 0 do 360 stepeni, a brzina kretanja je od 0 (sprajt stoji) do 15 (sprajt se kreće najvećom brzinom). Kada startujemo ovu naredbu sprajt je započeo svoje kretanje i ono će trajati sve dok ne zadamo neku drugu naredbu koja bi ga zaustavila. Pomoću ove naredbe možemo iz BASIC-a kontrolisati kretanje svih osam sprajtova istovremeno. Dok se sprajtovi kreću po ekranu mi možemo pisati program ili, na primer, izvršiti neku grafičku funkciju. Na ovaj način možemo postići veoma lepe vizuelne efekte. Dok se svih osam sprajtova kreće po ekranu mi istovremeno možemo crtati krugove, kvadrate ili bojiti pojedina polja.

RSPPOS

Naredba koja je suprotna prethodnoj naredbi MOVSPR. Njen zadatak je da nam da informaciju o poziciji određenog sprajta kao i o brzini njegovog kretanja.

RSPPOS (broj parametar)

Iza naredbe u zagradi slede dva parametra. Prvi parametar određuje koji sprajt nas zanima i može biti u intervalu od 1 do 8. Drugi parametar može imati tri vrednosti, koje redom imaju sledeća značenja:

(0) dobijamo informaciju o poziciji sprajta po X-osi (0—511),
(1) dobijamo informaciju o poziciji sprajta po Y-osi (0—255),
(2) dobijamo informaciju o brzini kretanja sprajta (0—15).

Najbolje je vrednost ove funkcije dodeliti nekoj numeričkoj promenljivoj, na primer

```
P=RSPPOS (1,2)
```

Dalje, u zavisnosti od promenljive P, vršiti grananje programa.

COLLISION

Postavlja se pitanje kako ustanoviti da li su se dva sprajta dodirnula, ili da li je sprajt dodirnuo neki deo na ekranu. To možemo ustanoviti ako pratimo koordinate pozicije sprajta i stalno vršimo upoređivanje. Ali, kako to testiranje vršiti kada crtež na sprajtu ne zauzima ceo sprajt, već samo deo (na primer mala lopta) a zanima nas kada je ta lopta dodirnula drugi sprajt. I ovo možemo rešiti putem proračunavanja koordinata, ali je to mnogo komplikovano. Sve ove probleme rešava nam funkcija COLLISION.

```
COLLISION tip,broj linije
COLLISION tip
```

Kod ove funkcije razlikujemo dva oblika. Za oba oblika zajednički je parametar tip. On može imati tri vrednosti:

(1) detekcija dodira dva sprajta,
(2) detekcija dodira sprajta i crteža na ekranu,
(3) detekcija dodira sprajta i svetleće olovke *(light pen)*.

Prva naredba uključuje detektor dodira, dok ga druga isključuje. Kod prve imamo još jedan parametar, a to je broj programske linije. U slučaju kada je detektovan dodir, funkcija COLLISION se ponaša kao funkcija GOSUB. Program će otići na programsku liniju navedenu pod parametrom *broj linije* i izvršiti taj potprogram. Ovde je veoma važno napomenuti da funkcija RENUMBER ne menja broj programske linije kod naredbe COLLISION, tako da o tome moramo posebno voditi računa.

BUMP

Prethodna naredba detektuje dodir sprajta sa drugim sprajtom, crtežom na ekranu ili svetlećom olovkom, ali koji sprajt je prouzrokovao dodir — to ne znamo. Zato je uvedena funkcija BUMP. Ona daje informaciju koji je sprajt dodirnuo crtež na ekranu ili koja su se dva sprajta sudarila.

9*

BUMP (tip)

Iza funkcije u zagradi stoji parametar tip. On može imati samo dve vrednosti: 1 ili 2. Kada nas zanima detekcija dodira dva sprajta, stavićemo da parametar tip ima vrednost jedan, a kada želimo da ustanovimo koji je sprajt dodirnuo crtež na ekranu, stavićemo da parametar tip uzima vrednost 2. Vrednost koju dobijamo preko funkcije BUMP je u intervalu od 0 do 255. Svaki bit ovog broja pridružen je jednom sprajtu. Na taj način, kad dobijenu vrednost pretvorimo u binarni oblik, možemo zaključiti koja su se dva sprajta dodirnula. Ako su, na primer, sedmi i osmi bit setovani (jednaki jedinici), to znači da je došlo do dodira između sedmog i osmog sprajta. Iz sledeće tabele lako možemo ustanoviti koja su se dva sprajta dodirnula ili koji je sprajt dodirnuo crtež na ekranu:

Broj sprajta	BUMP vrednost	Binarna vrednost
1	1	00000001
2	2	00000010
3	4	00000100
4	8	00001000
5	16	00010000
6	32	00100000
7	64	01000000
8	128	10000000

Ako, na primer, dobijemo da je rezultat naredbe BUMP (1) jednak 9 (a to je 8+1=9), onda znamo da je došlo do dodira između prvog i četvrtog sprajta.

5.13. MUZIKA

Tonske mogućnosti Commodora 128 su potpuno iste kao i Commodora 64. Međutim, dok Basic C-128 ima šest naredbi koje u potpunosti podešavaju skoro sve tonske mogućnosti, Basic C-64 nema ni jednu naredbu koja bi podržavala rad sa muzikom. Zato celo ovo poglavlje posvećujemo isključivo Commodoru 128 i sve naredbe koje budemo obrađivali važe samo za Basic V 7.0.

SID čip, koji se nalazi u Commodore-u ima mogućnost da kontroliše tri potpuno nezavisna ton generatora. Ton možemo dobiti preko televizora, tj. antenskim putem, ili na kompozitorskom izlazu za monitor. Ton dobijamo samo preko dve žice, i to MASE i AUDIO OUT izlaza. To znači da ton generatore ne možemo odvojiti na izlazu iz kompjutera niti možemo dobiti stereo zvuk. Ako želite da dobijete što čistiji ton, preporučujemo vam da ga ne koristite na televizoru, već da signal uzmete sa izlaza na kompozitnom monitoru. Ton dobijate preko drugog (masa) i trećeg (AUDIO OUT) pina na ovom džeku. Izlaz sprovedite ka pojačalu i na taj način dobićete najčistiji ton.

Pre nego što počnemo sa objašnjavanjem svake funkcije ponaosob, napomenućemo da je raspon tona svakog ton-generatora sedam oktava

Izbor oktave jednog ton-generatora nema nikakav uticaj na izbor oktave nekog drugog ton-generatora. Na ovaj način možete iz Basic-a dobiti melodije sa pratnjom ili istovremeno simulirati više instrumenata.

VOL

Prva funkcija koju obrađujemo ima zadatak da reguliše jačinu zvuka.

VOL n

Iza nje sledi parametar *n* koji može biti u intervalu od 0 do 15. Njegov zadatak je da odredi jačinu dobijenog tona. Kada je njegova vrednost 0, ton je potpuno isključen, a za vrednost 15 dobijamo maksimalnu jačinu. Ova funkcija obavezno ide uz naredbu SOUND, dok za naredbu PLAY nije neophodna. Jačinu tona možemo regulisati i potenciometrom na televizoru ili pojačalu. Međutim, ako je potrebno da se u toku neke melodije menja jačina zvuka, onda je ova naredba neophodna. Na primer: VOL 12

SOUND

Ova naredba služi za proizvođenje različitih tonskih efekata. Pomoću nje možemo dobiti tonove različitih frekvencija, dužina trajanja, oscilujuće tonove...

SOUND broj,fr,trajanje,način,minFr,korak,oblik,puls

Krenimo redom da vidimo koja je uloga svakog od navedenih parametara u sintaksi naredbe. Da odmah napomenemo da za korektno korišćenje ove funkcije moramo navesti prva tri parametra. Ostale po potrebi možemo ali i ne možemo koristiti.

broj. Određuje koji ton-generator koristimo. S obzirom na to da na raspolaganju imamo u tri ton-generatora, vrednost ovog parametra može biti 1,2 ili 3. Možemo navesti tri SOUND funkcije jednu za drugom i u svakoj definisati drugi ton-generator. Kada startujemo ovakav program, dobićemo harmonik koji se sastoji od dva ili tri potpuno različita tona. Možemo primeniti i različite efekte sa tonovima koje obrađuju naredni parametri i tako dobiti željeni efekat.

fr (frekvencija). Ovaj parametar određuje frekvenciju proizvedenog tona. Njegova vrednost može biti od 0 do 65535. Jasno je: što je ovaj parametar niži dobijamo dublji ton, a što ima veću vrednost, proizvedeni ton je viši.

trajanje. Ovo je poslednji parametar koji mora biti naveden kod ove naredbe. Znači, moramo da odredimo koji ton-generator koristimo, zatim kolika je frekvencija tona i, na kraju, kolika je dužina dobijenog tona. Ovaj parametar može biti u intervalu od 0 do 32767. Ako nas za-

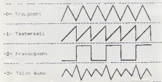
nima kolika je dužina tona u sekundama, dovoljno je da vrednost ovog parametra podelimo sa 60 i dobićemo željenu dužinu. To znači da minimalna dužina tona iznosi 1/60 (0.02) sekunde, a maksimalna dužina tona 32767/60 (546.12) sekundi (ili približno 9 minuta).

način. Ovaj parametar određuje način trajanja tona. On je povezan sa sledeća dva parametra i može imati tri vrednosti: 0,1 ili 2. Za vrednost 0 ton raste, tj. ide od niže ka višoj frekvenciji. Za vrednost 1 ton opada (od više ka nižoj frekvenciji), a za vrednost 2 ton osciluje između ove dve frekvencije. Ako ovaj parametar nije zadat, podrazumeva se da je njegova vrednost nula.

minFr (minimalna frekvencija). Kada koristimo prethodni parametar (način), ovaj parametar dolazi do izražaja. Parametrom (fr) smo odredili glavnu frekvenciju tona. Ovim određujemo minimalnu frekvenciju, a prethodni parametar određuje da li će ton ići od niže ka višoj frekvenciji, od više ka nižoj frekvenciji, ili će oscilovati između ove dve frekfencije. Kao i parametar (fr) i ovaj se nalazi u intervalu od 0 do 65535. Pošto ovaj parametar nije neophodno navoditi, ako to ne učinimo podrazumeva se da je njegova vrednost nula.

korak. Prethodna dva parametra određuju raspon frekvencija i da li ton raste, opada ili osciluje. Ovaj parametar određuje korak promene frekvencije. Ako, na primer, parametar (način) ima vrednost 0, znači da ton raste od niže ka višoj frekvenciji. Da li će za zadato vreme ton dospeti do maksimalne frekvencije ili neće određuje parametar (korak). Ako je vrednost parametra (korak) veliki broj, može se desiti da ton više puta pređe vrednosti od niže ka višoj frekvenciji. Inače ovaj parametar može imati vrednost u intervalu od 0 do 32767, a ako nije zadat podrazumeva se da je njegova vrednost jednaka 0.

oblik (oblik talasa). Ovaj parametar može imati četiri vrednosti i to od 0 do 3. U zavisnosti od njegove vrednosti imamo sledeće oblike talasa:



Trougaoni talas ima nežan i nešto tiši ton. Testerasti i pravougaoni oblik talasa daju oštriji i jači ton. Talas oblika šuma je nemuzički ton i koristi se za posebne efekte. Ako ovaj parametar nije zadat, podrazumeva se da ima vrednost 2.

puls Poslednji parametar kod funkcije SOUND odnosi se na boju tona. On ima uticaja samo ako smo izabrali pravougaoni oblik talasa (oblik 2), dok kod ostalih oblika on nema nikakvog uticaja. Kod pravougaonog oblika talasa možemo razlikovati gornju i donju stranu pravougaonika. Odnos dužina ovih stranica određuje ovaj parametar. On može imati vrednost u intervalu od 0 do 4095. Pošto nije neophodno da zadamo ovaj parametar, ako to nismo učinili on automatski uzima vrednost 2048.

Primeri.

SOUND 1,10000,60 koristi prvi ton-generator i proizvodi ton frekfencije 10000 u trajanju od jedne sekunde.

SOUND 2,12000,120,0,2000,100 — koristimo drugi ton-generator. Frekvencija raste od 2000 do 12000 sa korakom 100. Vreme trajanja je 120/60=2 sekunde.

SOUND 3,20000,600,2,2000,500 — dobićemo ton u trajanju od 10 sekundi koji pulsira između frekvencije 2000 i 20000 sa korakom 500.

PLAY

Ovu naredbu koristimo za pravljenje kompozicija na kompjuteru. Iza naredbe postoji samo jedan parametar, i to alfanumerička promenljiva. Umesto da niz karaktera između znaka navoda dodelimo alfanumeričkoj promenljivoj a tu promenljivu PLAY funkciji, možemo jednostavno iza naredbe PLAY pod znacima navoda navesti sve karaktere koji definišu kompoziciju. Mnoga slova koja koristimo pod znacima navoda za funkcije PLAY imaju posebno značenje.

PLAY promenljiva$

Sada ćemo redom navesti sve vrednosti koje može da postigne parametar *promenljiva* i objasniti njihovo dejstvo.

"Vn" — Slovo V određuje ton-generator koji koristimo. Iza njega je parametar 'n' koji može imati vrednosti 1,2 ili 3 — u zavisnosti od ton-generatora koji želimo da koristimo. Na primer, PLAY "V2" znači da koristimo drugi ton-generator.

"On" — Slovo O određuje oktavu u kojoj će biti sledeći tonovi. Iza slova je parametar 'n' koji može imati vrednosti u intervalu od 0 do 6. To znači da možemo proizvesti tonove u rasponu od 7 oktava. Ako slovo O ne navedemo, podrazumeva se da radimo u četvrtoj oktavi. Na primer PLAY "O3" znači da ćemo dobiti tonove iz treće oktave.

"Tn" — Slovom T biramo instrument koji želimo da imitiramo. Parametar 'n', koji može biti u intervalu od 0 do 9, određuje koji instrument imitiramo. Iz sledeće tabele možemo videti zavisnost parametra 'n' i vrste instrumenta.

n	Instrument
0	klavir
1	harmonika
2	kaliopa
3	bubanj
4	flauta
5	gitara
6	čembalo
7	orgulje
8	truba
9	ksilofon

Ako vrednost ovog parametra ne navedemo, podrazumeva se da je njegova vrednost 0, tj. da imitiramo klavir. Na primer, PLAY "T5" znači da želimo imitirati zvuk gitare.

"Un" — Parametar 'n' iz slova U određuje jačinu tona. Jačinu takođe možemo odrediti funkcijom VOL, ali u tom slučaju ona je ista za sva tri ton-generatora. Na ovaj način možemo imati različitu jačinu tona kod različitih ton-generatora. Vrednost parametra 'n' mora se nalaziti u intervalu od 0 do 15. Ni ovaj parametar nije neophodno navoditi, pa ako to ne učinimo — podrazumeva se da je njegova vrednost 9. Ako otkucamo PLAY "U15", to znači da želimo da naš ton bude maksimalne jačine.

"Xn" — vrednost parametra 'n' koji je iza slova X može imati samo dve vrednosti (0 ili 1). On služi za uključivanje ili isključivanje filtera. Ako je njegova vrednost 0 — filteri su isključeni, a ako pak ima vrednost 1 — filtri su uključeni. Ako vrednost ovog parametra ne navedemo podrazumeva se da je 0. Na primer, PLAY "X1" znači da su filtri uključeni. O filtrima ćemo govoriti u funkciji FILTER.

'A B C D E F G' — Prvih sedam slova abecede određuju note. Jednu oktavu određuju redom sledeće note 'CDEFGAB' (Do Re Mi Fa So La Si). Pišući ova slova na ovaj način — pravimo kompoziciju. Razmake između slova nije neophodno pisati. U početku vam preporučujemo radi lakše preglednosti. Na primer, PLAY "CDEFGAB" odsviraće notnu lestvicu u četvrtoj oktavi (oktava nije navedena, pa automatski uzima vrednost 4).

"#" — Ako se ovaj znak nalazi ispred slova za notu, ona će biti povišena za pola tona. U primeru PLAY "C #C D" — dobićemo tri tona poređana po visini.

"$" — Znak suprotan prethodnom. Kada se ovaj znak nalazi ispred slova za notu, ona će biti snižena za jednu polovinu. Tako, u primeru PLAY "C $D D" dobićemo isti efekat kao u prethodnom slučaju. U prethodnom primeru je nota C povišena za pola tona, a ovde je nota D snižena za pola tona.

Sada dolazi skup slova koji određuje dužinu trajanja nota. Iz donje tabele može se videti koje slovo određuje kokiko će trajati dužina note:

Slovo	Dužina note
S	šesnaestina (1/16)
I	osmina (1/8)
Q	četvrtina (1/4)
H	polovina (1/2)
W	cela nota (1/1)
.	produžena cela nota
M	završni takt
R	pauza

Potrebno je da ispred slova za notu ili ispred skupa slova postavimo znak za dužinu note. Sve do sledećeg znaka za dužinu note će trajati koliko smo odredili. Na primer PLAY "SC WE C ID", nota C će trajati jednu šesnaestinu, zatim note E i C po jednu celu notu, i na kraju nota D će trajati jednu osminu.

Ovo bi bili svi parametri koji se mogu primeniti kod PLAY funkcije. Napomenimo još da u okviru jedne PLAY naredbe možemo koristiti više ton-generatora. Na primer: PLAY "V1 C V2 D V3 E". Kod ovog konkretnog primera imamo harmoniju koja se sastoji od tri tona (C, D i E). Pošto za svaki ton koristimo po jedan ton-generator, oni će se čuti istovremeno.

TEMPO

Sledeća funkcija nam koristi za određivanje brzine izvođenja neke kompozicije:

TEMPO vreme

Ova naredba sama za sebe ne proizvodi nikakav ton, već je pridodata PLAY naredbi. Parametar (vreme) koji sledi za naredbe može imati vrednost iz intervala od 0 do 255. Ako ne navedemo ovu naredbu, podrazumeva se da je vreme osam. Ako vrednost parametra (vreme) povećamo, kompjuter će brže svirati. Najbolje ćete shvatiti uticaj ove naredbe na sledećem primeru:

```
10 TEMPO 5: PLAY "CDEFGAB"
20 TEMPO 50:PLAY "CDEFGAB"
```

Razlika u brzini više je nego očigledna.

ENVELOPE

Ova naredba nam koristi za definisanje oblika talasa — da bismo što vernije mogli da imitiramo neki instrument. Ona sama ne može da proizvede nikakav zvuk, već se pridodaje PLAY naredbi. Kada je primenjujemo, jedino prvi parametar moramo da napišemo, a ostale parametre možemo, ali ih i ne moramo pisati.

ENVELOPE broj,vp,vs,jačina,vu oblik puls

Sada krenimo redom da vidimo koju ulogu ima koji parametar. Prvi parametar (broj) može imati vrednosti od 0 do 9 i njegova uloga je da odredimo instrument koji želimo da imitiramo. Iz sledeće tabele možemo videti koju vrednost parametra treba izabrati da bismo dobili zvuk željenog instrumenta.

Broj	Instrument
0	klavir
1	harmonika
2	kaliopa
3	bubanj
4	flauta
5	gitara
6	čembalo
7	orgulje
8	truba
9	ksilofon

Ako na primer želimo da imitiramo zvuk trube, napisaćemo sledeće: ENVELOPE 8

vp (vreme pojačanja) — Ovim parametrom određujemo vreme koje je potrebno da ton dostigne maksimalnu jačinu. Vrednosti koje ovaj parametar može da ima su u intervalu od 0 do 15.

vs (vreme smanjenja) — Ovim parametrom određujemo vreme koje je potrebno da ton sa maksimalne jačine dođe do jačine koja je određena sledećim parametrom (jačina). Kao i kod prethodnog parametra, i ovaj može uzimati vrednosti iz intervala od 0 do 15.

jačina — Ovaj parametar određuje srednju jačinu tona. Dok vrednosti prethodna dva parametra i parametra koji sledi iza ovog čitamo na X-osi, vrednost ovog parametra se čita na Y-osi (priloženi grafikon). Pošto ovaj parametar određuje jačinu, on može biti u intervalu od 0 do 15.

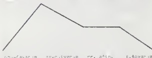
vu (vreme utišavanja) — Pomoću ovog parametra određujemo vreme koje je potrebno da se ton čija je jačina definisana prethodnim parametrom potpuno utiša. Vrednosti koje ovaj parametar može da uzme su iz intervala od 0 do 15.

Oblik — Ovaj parametar određuje oblik talasa. On može imati vrednost od 0 do 4. U zavisnosti od vrednosti imamo sledeće oblike talasa.

(0) trougaoni
(1) testerasti
(2) pravougaoni
(3) oblik šuma
(4) zvuk zvona

Parametri od 0 do 3 određuju potpuno isti oblik talasa kao i naredba SOUND. U opisu te naredbe može se videti grafički prikaz oblika tih talasa.

puls — Poslednji parametar kod funkcije ENVELOPE odnosi se na boju tona. On ima uticaja samo ako smo izabrali pravougaoni oblik talasa (oblik 2). Kod ostalih oblika on nema nikakvog uticaja. Kod pravougaonog oblika talasa možemo razlikovati gornju i donju stranu pravougaonika. Odnos dužina ovih stranica određuje ovaj parametar. On može imati vrednost u intervalu od 0 do 4095.



To bi bili svi parametri koji poseduje funkcije ENVELOPE. Pošto je jedino prvi parametar neophodan, pretpostavljate da ostali parametri automatski dobijaju neku vrednost. Pošto ta vrednost zavisi od prvog parametra, ovde ćemo dati tabelu iz koje se mogu videti sve ostale vrednosti koje kompjuter automatski uzima ako nisu navedene.

broj	instrument	vp	vs	jačina	vu	oblik	puls
0	klavir	0	9	0	0	2	1536
1	harmonika	12	0	12	0	1	—
2	kaliopa	0	0	15	0	0	—
3	bubanj	0	5	5	0	3	—
4	flauta	9	4	4	0	0	—
5	gitara	0	9	2	1	1	—
6	čembalo	0	9	0	0	2	512
7	orgulje	0	9	9	0	2	2048
8	truba	8	9	4	1	2	512
9	ksilofon	0	9	0	0	0	—

Ako vam ove vrednosti ne odgovaraju, onda ih u funkciji ENVELOPE navedite i one će biti promenjene.

FILTER

Poslednja funkcija koju određujemo u ovom poglavlju daje posebnu boju tona. Pomoću naredbi PLAY i ENVELOPE možemo ostvariti željenu kompoziciju i imitirati određeni instrument. Međutim, ako tonom nismo u potpunosti zadovoljni, možemo ga korigovati pomoću naredbe FILTER i tako stvarno postići željeni ton.

FILTER fr,nizak,srednji,visok rezonansa

Uloga ove funkcije je da filtrira tonove koje proizvodimo PLAY naredbom. Iza nje slede parametri koje ćemo sada redom objasniti.

fr — Ovim parametrom određujemo frekvenciju koja se koristi kao osnovna frekfencija za filtriranje tonova. Parametar može biti u intervalu od 0 do 2047.

nizak — Ovaj parametar može imati samo dve vrednosti (0 ili 1). Ako je nula, to znači da je ova filtracija isključena, a ako je pak 1, onda naredba FILTER ima ulogu da propušta samo tonove koji su ispod frekfencije određene prvim parametrom.

srednji — I ovaj parametar može imati samo dve vrednosti (0 ili 1). Kao i prethodni u slučaju da je jednak nuli — isključuje se ovaj način filtriranja, a ako je jednak jedinici — funkcija FILTER propušta samo tonove koji su bliski frekfenciji zadatoj prvim parametrom.

visok — Kao i prethodna dva parametra i ovaj može imati samo dve vrednosti (0 ili 1). U slučaju da je njegova vrednost 0, ovaj mod filtriranja nije uključen, a ako je pak jednak jedinici, funkcija propušta samo tonove koji su iznad frekfencije određene prvim parametrom.

rezonansa — Ovo je poslednji parametar koji može imati vrednosti u intervalu od 0 do 15. Od njega zavisi propusna moć filtera. Ako funkcija FILTER ima zadatak da propušta samo niske tonove, to znači da će visoki tonovi biti prigušeni, a koliko će oni biti prigušeni — određuje ovaj parametar.

Sada ćemo dati jedan primer gde možete čuti razliku između tonova sa i bez filtera, kao i razliku kada se koriste pojedine vrste filtera.

```
10 TEMPO 8
20 PLAY "O3 T0 U12 HE F E D WC"
30 PRINT CHR$(147) "PRITISNITE BILO KOJE DUGME"
40 FILTER 1024,1,0,0,7
50 PLAY "X1 HE F E D WC"
```

Nakon ovoga promenite program tako što ćete umesto linije 40 napisati

```
40 FILTER 1024,0,1,0,7
```

a posle izvršite sledeću izmenu

```
40 FILTER 1024,0,0,1,7
```

Ovo bi predstavljalo svih šest funkcija koje su vezane za muziku kod C-128. Kao što ste i sami mogli videti, programi pisani u basic-u mogu proizvesti željene tonove i odsvirati razne kompozicije. Da napomenemo da su tonske mogućnosti C-128 još veće, ali za to je potrebno pisanje programa u mašinskom jeziku ili korišćenje nekog komercijalnog programa za muziku. U modu 64 takvih programa ima zaista dosta. Demonstracije pojedinih programa mogu izazvati čuđenje koliko verno kompjuter može da imitira određene instrumente.

5.14. MONITOR

COMMODORE 128 u svom operativnom sistemu ima skup naredbi za rad u mašinskom jeziku. Prelazak u ovaj mod postiže se Basic naredbom MONITOR. U ovoj knjizi nećemo obrađivati mašinski jezik, već ćemo samo predstaviti sve funkcije koje se nalaze u okviru naredbe MONITOR. Pomoću skupa ovih naredbi možemo pisati ili listati neki mašinski program. Na raspolaganju imamo 21 funkciju i ovde ćemo ih sada predstaviti.

Asembler

A adresa naredba operator

Slovo A je funkcija MONITORA koja služi za pisanje mašinskih programa. Sintaksa naredbe je sledeća: iza slova A dolazi adresa u heksadekadnom brojnom sistemu. Adresu je neophodno napisati kako bi kompjuter znao u kom delu memorije želimo da pišemo naš program. Adresa se sastoji od pet karaktera, od kojih prvi određuje banku podataka u kojoj pišemo program, a ostala četiri memorijsku lokaciju. Ako ne napišemo banku u kojoj radimo, već samo adresu od četiri karaktera, podrazumeva se da radimo u nultoj banci.

Iza adrese dolazi naredba mašinskog jezika, a ako ona zahteva i operator — moramo ga napisati. Nakon toga pritiskamo taster (RETURN) i kompjuter će automatski sračunati koja je sledeća memorijska lokacija, ispisati je i biti spreman da prihvati novu mašinsku naredbu. Pogledajmo kako sve to izgleda na jednom primeru:

```
A 1300 LDA #$00
```

Otkucali smo A što znači da želimo da pišemo program u asembleru. Iza toga smo napisali broj 1300. Pošto on zauzima četiri karaktera, to znači da nismo definisali banku podataka u kojoj radimo, pa kompjuter podrazumeva da radimo u nultoj. Nakon toga sledi naredba LDA koja znači da akumulator treba da uzme određenu vrednost. Iza nje pišemo #$00 što predstavlja operator, tj. akumulator uzima vrednost nula. Kada pritisnemo (RETURN) na ekranu ćemo dobiti:

```
A01300 A9 00      LDA #$00
A01302
```

Kompjuter će automatski napisati kodove ove naredbe a ispod zadate adrese napisaće novu i biće spreman da prihvati sledeću mašinsku instrukciju. Na ovaj način možemo pisati mašinski program.

Upoređivanje

C adresa1 adresa2 adresa3

Ova naredba služi za pronalaženje razlika u sadržaju između dva memorijska bloka podataka. Iza slova C potrebno je da napišemo adresu početka bloka podataka koji nas zanima za upoređivanje. Zatim pišemo krajnju adresu tog bloka i na kraju, zadajemo početnu adresu novog bloka sa kojim vršimo upoređivanje. Za svaku adresu potrebno je da napišemo po pet karaktera od kojih prvi određuje banku u kojoj radimo. U slučaju da smo napisali samo četiri karaktera, podrazumeva se da radimo u nultoj banci. Na primer, ako želimo da pronađemo sve razlike između blokova podataka od x2000 do $3000 u nultoj banci i bloka u prvoj banci, otkucaćemo sledeće

C 2000 3000 12000

i kompjuter će nam ispisati sve razlike između ova dva bloka. U slučaju da ne napiše ni jedan broj znači da su blokovi podataka potpuno isti.

Disasembler

D adresa1 adresa2

Da bismo mogli da analiziramo neki mašinski program, neophodno je da imamo njegov listing. U tu svrhu koristićemo funkciju D koja nam omogućuje da skup brojeva koji sačinjavaju mašinski program pretvori u listing. Iza slova D treba da napišemo početnu i krajnju adresu programa koji želimo da izlistamo. Adresa se sastoji od pet karaktera. Prvi karakter određuje banku podataka u kojoj se nalazi program čiji listing želimo da dobijemo, a ostala četiri karaktera određuju memorijsku lokaciju programa u toj banci. Prvi karakter nije neophodno navoditi, pa u slučaju da ga nismo napisali — podrazumeva se da radimo u nultoj banci. Takođe nije neophodno navesti adresu-2. Ako je ne navedemo, dobićemo listing čija je dužina 21 bajt. Ako želimo da dalje listamo program, dovoljno je da pritisnemo D i (RETURN). Tako ćemo dobiti sledeći listing koji je takođe dug 21 bajt. I tako, redom, možemo izlistati ceo program. Na primer, ako otkucamo

D FE000

na ekranu ćemo dobiti

```
. FE000 A2 FF       LDX #$FF
. FE002 78          SEI
. FE003 9A          TXS
. FE004 D8          CLD
. FE005 A9 00       LDA #$00
. FE007 8D 00 FF    STA $FF00
. FE00A A2 0A       LDX #$0A
. FE00C BD 4B E0    LDA $E04B,X
. FE00F 9D 00 D5    STA $D500,X
. FE012 CA          DEX
. FE013 10 F7       BPL $E00C
```

U ovom primeru smo izlistali deo operativnog sistema koji se nalazi u banci 15 od adrese $E000

Popunjavanje memorije

F adresa1 adresa2 bajt

Ova naredba služi da određeni deo memorije popunimo datim brojem. Za nju je potrebno da navedemo tri parametra, i to sva tri obavezno moramo navesti. Prvi i drugi parametar određuju adresu početka i kraja memorijskog bloka koji popunjavamo. Adrese se sastoje od po pet karaktera. Prvi karakter određuje banku podataka u kojoj se nalazi deo memorije, a ostala četiri početnu, tj. krajnju adresu dela memorije. Sledeći parametar bajt je broj kojim popunjavamo memoriju. On mora biti u intervalu od 0 do 255. Pri korišćenju ove funkcije vodimo računa o adresama koje koristimo. Može se desiti, ako zakačimo sistemske promenljive, da kompjuter ostane blokiran. Ako otkucemo sledeći primer

F 13000 14000 00

deo memorije od adrese $3000 do $4000 u prvoj banci podataka biće popunjen nulama.

Startovanje programa

G adresa

Poznato je da BASIC naredba SYS (adresa) startuje mašinski program. Takođe i iz monitora možemo startovati mašinski program koji se nalazi u bilo kom delu memorije. Naredba ima samo jedan parametar — to je startna adresa. Kao i kod prethodnih funkcija ona se sastoji od pet karaktera, od kojih prvi karakter predstavlja banku podataka, a ostala četiri adresu programa koji želimo da startujemo. Na primer, otkucajte sledeće

G FE000

Manifestacija koju ste dobili je potpuno ista kao da ste pritisnuli reset taster. U 15. banci na adresi $E000 je start programa nakon reseta

Pretraživanje memorija

H adresa1 adresa2 podaci

Kada biste hteli da u nekom mašinskom programu pronađete određene instrukcije listajući program, ubrzo biste odustali. U tu svrhu u okviru MONITORA postoji naredba za automatsko pretraživanje programa. Možete tražiti određene kodove naredbe mašinskog jezika ili neki tekst. Iza slova H potrebno je da navedete početnu i krajnju adresu programa. Adresa se sastoji od pet karaktera. Prvi karakter je banka podataka u kojoj se nalazi program, a ostala četiri određuju adresu unutar te banke. Ako se ne navede banka podataka, kompjuter podrazumeva da se program nalazi u nultoj banci. Poslednji parametar kod ove funkcije podrazumeva brojeve ili tekst. Ako želimo da pronađemo neki tekst, njega moramo navesti između znakova za apostrof (’). Na primer

H 4000 5000 A9 00

ili ako tražimo tekst

H 12000 15000 'DEMO

U drugom primeru vršimo pretraživanje prve banke podataka od adrese \$2000 do \$5000 i tražimo reč DEMO. Kompjuter će ispisati sve adrese gde je pronašao tekst ili tražene brojeve. Ako ne napiše nijednu adresu, znači da traženi tekst ili brojevi ne postoje u tom delu memorije.

Učitavanje

L'ime programa',broj,adresa

Ova naredba je veoma slična Basic naredbi LOAD. Služi za učitavanje programa ili bloka podataka u memoriju računara. Učitavanje se vrši sa kasetofona ili diska. Takođe možemo odrediti banku podataka i adresu gde želimo da učitamo naš program. Iza slova L, pod znacima navoda, moramo da napišemo ime programa koji želimo da učitamo. Zatim dolazi broj 1 — za učitavanje sa kasetofona ili broj 8 — ako želimo da program učitamo sa diska. Ako ne stavimo parametar adresa, program će se učitati na originalno mesto, tj. na mesto sa koga je snimljen na disk ili kasetofon. Ako, pak, navedemo adresu stavimo, program će se učitati na željeno mesto i to u banku podataka koju smo mi izabrali.

L'DEMO 11, 8 10800

Dati primer će učitati program pod nazivom DEMO 11 sa diska u prvu banku podataka počevši od adrese \$0800

Listanje kodova

M adresa1 adresa2

Za listanje mašinskih programa koristili smo funkciju D. Ako pak želimo da izlistamo neke podatke koji nisu program, takođe možemo koristiti funkciju D. U tom slučaju mašinski listing neće imati nikakvog smisla. Zbog toga koristimo novu funkciju koja ima oznaku M. Pomoću nje na ekranu možemo dobiti brojeve koji se nalaze u memoriji kao i ASC kodove. Iza naredbe dolaze dva parametra: početna i krajnja adresa bloka podataka koji želimo da pregledamo. U slučaju da ne koristimo drugu adresu, tj. kraj bloka podataka, na ekranu će se pojaviti prva 192 broja bloka podataka. Za dalje listanje dovoljno je da otkucamo M i (RETURN) i dobićemo sledeća 192 broja. To sve važi ako koristimo izlaz preko RGB monitora.

U slučaju da koristimo kompozitni monitor ili televizor, sve je potpuno isto osim broja podataka koje ćemo dobiti ako ne koristimo drugu adresu. U ovom slučaju dobićemo prvih 96 brojeva. Za nastavak listanja takođe otkucamo M i (RETURN) i dobićemo drugih 96 brojeva. Pored ovih brojeva dobijamo i ASC kodove. Pomoću njih možemo veoma lako pročitati tekst koji se nalazi u tom delu memorije. Na primer, ako otkucamo

```
M 1000 1018
```

dobićemo sledeće podatke

```
>01000 07 06 0A 07 06 04 05 08
>01008 09 05 47 52 41 50 48 49    GRAPHI
>01010 43 44 4C 4F 41 44 22 44 CDLOAD"D
>01018 49 52 45 43 54 4F 52 59 IRECTORY
```

Iz primera se vidi način na koji dobijamo brojeve i tekst na ekranu.

Registri

R

Ova naredba iza sebe ne sadrži nikakav parametar. Koristimo je za prikaz sadržaja registra i stanja programskog brojača. Ako nakon izvršenja nekog mašinskog programa otkucamo R i (RETURN), dobićemo sledeće podatke:

PC — stanje programskog brojača, tj. na kojoj adresi se završio mašinski program

SR — stanje flagova mikroprocesora

AC — vrednost koja se nalazi u akumulatoru.

XR — vrednost koja se nalazi u X-registru

YR — vrednost koja se nalazi u Y-registru

SP — pokazivač stek memorije

Na primer:

```
R
  PC      SR AC XR YR SP
,03005    12 23 45 00 F8
```

Iz gornjeg primera vidimo sledeće podatke: programski brojač se zaustavio na adresi $3005 u nultoj banci, vrednost flagova je 12. Da bismo videli koji flag ima koju vrednost, moramo broj 12 pretvoriti u binarni oblik. Dalje, akumulator, X i Y registri imaju redom sledeće vrednosti: 23,45 i 0. Na kraju imamo podatak da je ukazivač stek memorije F8.

Snimanje programa

S"ime programa",broj,adresa1,adresa2

Naredba koja je potpuno ista kao i Basic-naredba BSAVE (block save). Iza slova S potrebno je da između navoda napišemo ime programa. Zatim dolazi broj periferije na koju snimamo program. Kad želimo da snimimo na disk — broj je osam, a za kasetofon broj mora biti jedan. Nakon toga dolazi početna i krajnja adresa programa. Obe adrese se sastoje od po pet karaktera. Prvi karakter određuje banku podataka u kojoj se nalazi program, a ostala četiri početnu odnosno krajnju adresu programa. Da bismo korektno snimili program moramo u naredbi krajnju adresu programa uvećati za jedan. Na primer:

S"DEMO 9",8,04000 05001

Ovaj konkretan primer će na disk snimiti program pod nazivom demo 9 koji se nalazi u nultoj banci. Početna adresa programa je $4000 a krajnja $5000. Kao što vidimo, da bismo ceo program korektno snimili moramo krajnju adresu uvećati za jedan.

Kopiranje bloka memorije

T adresa1 adresa2 adresa3

Ova funkcija služi za prebacivanje određenog dêla memorije. Prebacivanje se može vršiti unutar jedne banke podataka, a možemo prebacivati i podatke iz jedne u drugu banku podataka. Iza naredbe obavezno moramo navesti tri podatka. Prvo početnu adresu dêla memorije koji želimo da prebacimo, zatim krajnju adresu i, na kraju, početnu adresu dêla memorije gde želimo da prebacimo podatke. Sva tri parametra zauzimaju po pet karaktera, od kojih prvi određuje banku podataka, a ostala četiri adrese. Na primer, ako otkucamo

T 3000 4000 17000

kompjuter će prebaciti deo memorije od adrese $300 do $4000 iz nulte banke podataka (jer nije navedena banka) u prvu banku, počevši od adrese $7000.

Ovu funkciju možemo koristiti i za prebacivanje programa. Samo — vodite računa o programima koji u sebi sadrže direktne skokove, jer ova naredba ne menja te skokove. Zato prebačeni program mora mo ručno izmeniti — da bi radio na novom mestu.

Varifikacija

V"ime programa",broj,adresa

Naredba koja je veoma slična Basic-naredbi VERIFY. Kada smo snimili neki program na disk ili kasetofon, pa nas zanima da li je program dobro snimljen, možemo koristiti ovu funkciju. Iza slova V, pod znacima navoda, pišemo ime programa, zatim broj periferije (1 za kasetofon ili 8 za disk) i,na kraju, adresu programa. Ovu naredbu možemo koristiti i za proveru da li je došlo do promene kod nekog programa nakon njegovog startovanja. Na primer, snimili smo neki program ili podatke na disk, pa želimo da izvršimo proveru da li su dobro snimljeni. Otkucaćemo sledeće:

V"DEMO 3",8

Nakon ovoga disk će se zavrteti i kompjuter će početi sa proverom. U slučaju da je sve u redu, dobićemo poruku VERIFY OK, a u slučaju greške kompjuter će saopštiti odgovarajuću poruku. Ako je došlo do greške program moramo ponovo snimiti.

Povratak u Basic

X

Kada smo završili sa radom u MONITORU i želimo da se vratimo u Basic otkucaćemo X i (RETURN). Kompjuter će napisati poruku READY. Nakon toga možemo dalje nastaviti rad u Basic-u. Ako pokušamo da napišemo bilo koju naredbu iz MONITORA, dobićemo poruku o grešci.

Upisivanje brojeva u memoriju

>adresa bajt1 bajt2 ...

Ova naredba ima dvostruku ulogu. Pomoću nje možemo dobiti brojeve i ASC kodove određene adrese (slično kao kod funkcije M) a možemo unositi i podatke. Iza znaka > (znak za veće) obavezno moramo dati adresu koja se sastoji od 5 ili 4 karaktera, u zavisnosti od toga da li smo definisali banku podataka ili hoćemo da radimo u nul-

toj Ako ne napšemo nijedan parametar više, dobićemo sadržaj sledećih 16 bajta memorije na RGB monitoru, odnosno 8 bajta memorije na kompozitnom monitoru ili televizoru Ako pak iza adrese stavimo neki broj (jedan ili više) oni će biti smešteni na zadatoj adresi u memoriji Na primer ako želimo da brojeve 31 45 i FA postavimo na adresu $4500 $4501 i $4502 redom u prvoj banci podataka potrebno je da otkucamo

```
>14500 31 45 FA
```

Nakon pritiskanja tastera (RETURN) automatski dobijamo 16 ili 8 bajta memorije od adrese $4500, a takođe i njihove ASC vrednosti

Asembler

- adresa naredba operator

Ovu naredbu nećemo posebno obrađivati jer je potpuno ista kao i naredba 5 11 1 Uvedena je radi lakšeg ispravljanja već napisanog programa Pri korišćenju naredbe DISASEMBLER pre svake adrese postoji tačka (to se vidi iz primera) Kada želimo da promenimo neku instrukciju ili operator jednostavno otkucamo novu funkciju i pritisnemo (RETURN) Kompjuter će automatski izvršiti ispravku jer je tačku pre adrese shvatio isto kao funkciju A Pošto na raspolaganju imamo ekranski editor, a ispred svake adrese postoji tačka, veoma jednostavno i brzo možemo menjati program Kodove naredbe koji se nalaze između adrese i funkcije, ne moramo brisati niti menjati Kompjuter će ih automatski promeniti kada promenimo funkciju

Disk komanda

@ uređaj,naredba

Veoma korisna funkcija koja nam omogućuje da iz monitora radimo sa diskom Njena sintaksa je sledeća Odmah iza komanda kucamo broj diska. Ovaj parametar koristimo samo ako imamo dva ili više disk-drajeva U slučaju da imamo samo jednu disk-jedinicu, nije potrebno ni da ga navodimo jer ako ga nema — kompjuter automatski podrazumeva da je broj diska osam Nakon toga sledi zarez i disk komanda

U slučaju da ne navedemo nijedan parametar, već samo funkciju dobićemo trenutni status diska Na primer

```
@
00.OK,00,00
```

Ako, pak, želimo da vidimo sadržaj programa na disku — otkucaćemo sledeće

```
@ $
```

Na ekranu ćemo dobiti sadržaj diskete. Ovaj primer je potpuno isti kao Basic-funkcija DITECTORY. Dalje iza komande možemo otkucati bilo koju disk-komandu. Ako želimo da obrišemo neki program, pod imenom DEMO otkucaćemo

@ S0 DEMO

pa će program biti obrisan sa diska i dobićemo sledeću poruku

01,FILES SCRATCHED,01,00

Pored ovih funkcija možemo koristiti i mnoge druge, na primer Inicijalizacija, Formatizacija diska. Promena imena programa itd.

Promena registara mikroprocesora

, vrednosti

Pri korišćenju funkcije R (registri) možemo primetiti da ispred vrednosti programskog brojača stoji znak ;. Dalje, iza vrednosti programskog brojača stoje vrednosti registara mikroprocesora. Da bismo koristili ovu funkciju, potrebno je prethodno da otkucamo R, a nakon toga da odemo kursorom do željenog mesta i bilo koju vrednst registra mikroprocesora promenimo. Nakon toga pritisnemo (RETURN) i registri će automatski dobiti željene vrednosti.

Ovu funkciju najčešće koristimo ako želimo da vidimo kakav uticaj na registre i flegove ima neki mašinski program. Prvo postavimo željene vrednosti a nakon toga startujemo program. Kad se program zaustavi, na ekranu možemo videti sve promene do kojih je došlo prilikom korišćenja programa.

Ostale su nam još četiri funkcije koje imaju zadatak da pretvaraju brojeve iz jednog sistema u drugi. Možemo raditi sa četiri brojna sistema, i to

Heksadekadni	(osnova 16)
Dekadni	(osnova 10)
Oktalni	(osnova 8)
Binarni	(osnova 2)

Svaki brojni sistem ima određeni simbol koji se mora nalaziti ispred broja da bi kompjuter prepoznao sistem koji koristimo. Jedino pri korišćenju heksadekadnog brojnog sistema ne moramo navoditi predznak jer ako on ne postoji — kompjuter automatski podrazumeva da radimo u heksadekadnom brojnom sistemu. Sada ćemo redom obraditi svaki od navedenih brojnih sistema.

Heksadekadni brojni sistem

$ broj

Ovaj brojni sistem, čija je osnova 16, najviše se koristi. Pri svakom listanju mašinskih programa, adrese i kodovi su dati u ovom sistemu. Ako želimo da neki broj iz ovog sistema prebacimo u neki drugi brojni sistem, dovoljno je da otkucamo znak za dolar, zatim broj i (RETURN). Na primer, ako broj 1234, koji je dat u heksadekadnom brojnom sistemu želimo da prebacimo u drugi brojni sistem, otkucaćemo:

$1234

i dobićemo sledeći rezultat:

$1234
+4660
&11064
%1001000110100

Kao što se iz gornjeg primera vidi dobili smo ista vrednost u sva četiri brojna sistema. Prednaci imaju sledeću funkciju:

$	Heksadekadni brojni sistem
+	Dekadni brojni sistem
&	Oktalni brojni sistem
%	Binarni brojni sistem

Dekadni brojni sistem

+ broj

Dekadni brojni sistem koristimo u normalnom računu i svakodnevno ga primenjujemo. Pri pisanju Basic-programa koristimo ovaj brojni sistem. Ako želimo da iz Basic-a pozivamo neki mašinski program naredbom SYS (adresa), onda ćemo otkucati prednak + i željenu adresu. Na ekranu ćemo dobiti taj broj u sva četiri brojna sistema. Pogledaćemo koja mu je heksadekadna vrednost, i od te vrednosti ćemo početi sa pisanjem mašinskog programa. Na primer, ako otkucamo:

+50000

dobićemo:

$C350
+50000
&141520
%1100001101010000

Oktalni brojni sistem

& broj

Ovaj brojni sistem ima osnovu — osam. Pošto ga verovatno uopšte nećemo koristiti, nećemo ga posebno ni objašnjavati. Postupak pretvaranja broja iz oktalnog brojnog sistema u ostala je potpuno isti kao i kod ostalih brojnih sistema. Otkucamo predznak & i zatim oktalni broj. Na ekranu ćemo dobiti vrednost tog broja u ostalim brojnim sistemima.

Binarni brojni sistem

% broj

Poslednji brojni sistem koji ovde obrađujemo je binarni sistem čija je osnova 2. Njega ćemo vrlo često koristiti. Commodore 128 ima dosta registara čiji bitovi imaju određenu funkciju. Kako pri listanju memorije dobijamo samo heksadekadne vrednosti, moramo koristiti ovaj brojni sistem da bismo znali koji bit ima koju vrednost. Takođe ako želimo da znamo vrednost flagova u status-registru, moramo heksadekadnu vrednost pretvoriti u binarnu — da bismo znali tačno koji flag ima koju vrednost. Pored toga, ako želimo da neki bitovi registra budu setovani (imaju vrednost 1) a drugi ne — otkucaćemo predznak % i binarnu vrednost broja. Pritiskom na taster (RETURN) dobićemo tu vrednost u ostalim brojnim sistemima.

Na kraju da napomenemo: pri pretvaranju brojeva iz jednog u drugi brojni sistem postoji jedno ograničenje. Najveći heksadekadni broj koji možemo pretvarati u drugi sistem je $FFFFF, a najveći dekadni broj je +1048575.

DODATAK „A"

PORUKE O GREŠKAMA

Pri pisanju Basic-programa dešava se da pravimo različite greške. U takvim slučajevima kompjuter prekida izvršavanje programa i daje nam informaciju o vrsti greške koju smo napravili. Pritiskom na taster HELP možemo dobiti i programsku liniju u kojoj je načinjena greška.

U ovom prilogu dajemo sve poruke o greškama do kojih može doći pri radu sa Basic-programima. Svaka poruka je objašnjena tako da svoju grešku možete veoma lako otkloniti. Ako ste neku naredbu pogrešno otkucali, dobićete poruku o sintaksnoj grešci. Ili, ako neki niz pokušate da dimenzionišete, a to ste već prethodno učinili — kompjuter će takođe javiti da ste napravili grešku. Iza poruke o grešci u zagradi je dat broj greške. Taj broj koristimo primenom Basic-naredbi koje se odnose na greške (na primer TRAP i ER). Sada vam dajemo sve poruke o Basic-greškama na koje možete naići pišući neki program.

BAD DATA ERROR (greška bez broja). Ova greška se javlja jedino u modu C-64. Ista greška u modu C-128 je FILE DATA.

BAD DISK ERROR (greška broj 36). Poruku o ovoj grešci dobićemo pri formatizaciji diskete. Naredba za brzo formatizovanje diskete korektno radi na već formatizovanoj disketi, ali ako disketa nije formatizovana — dobićemo poruku o grešci. Takođe postoji mogućnost da je disketa fizički oštećena.

BAD SUBSCRIPT ERROR (greška broj 18). Kada koristimo nizove promenljivih, prethodno moramo te nizove dimenzionisati. U slučaju da niz nismo dimenzionisali, indeks niza može biti u intervalu od 0 do 10. Do ove greške dolazi jer je indeks manji ili veći od dozvoljenog broja. Indeks niza nipošto ne sme biti negativan broj.

BEND NOT FOUND ERROR (greška broj 37). Unutar IF THEN ELSE naredbi možemo da definišemo BEGIN BEND blok koji će biti izvršen ili posle THEN-a ili posle ELSE-a. Međutim, svaki BEGIN mora imati odgovarajući BAND. Na pojavu ove poruke proverite da li se svaka BEGIN naredba završava odgovarajućim BEND-om.

BREAK ERROR (greška broj 30) U suštini ovo nije greška, već poruka da smo neki Basic program prekinuli pritiskom na taster RUN/STOP Da bismo nastavili sa izvršavanjem programa dovoljno je da damo naredbu CONT.

CAN'T CONTINUE ERROR (greška broj 26). U slučaju da smo program zaustavili END ili STOP naredbom u programu, ili pritiskom na taster RUN/STOP — program možemo nastaviti Međutim, ako se program zaustavio zbog neke greške ili smo editovali neku programsku liniju — program ne možemo nastaviti i u tom slučaju ćemo dobiti ovu poruku o grešci Ako znamo gde je program stao možemo nastaviti njegovo izvršenje sa GOTO i programskom linijom prekida programa

CAN'T RESUME ERROR (greška broj 31) Naredba RESUME dozvoljava nastavak programa posle pojavljivanja greške, s tim što prethodno mora biti upotrebljena naredba TRAP Ovu poruku dobijamo ako smo upotrebili RESUME pre nego što je izvršena naredba TRAP Preporučujemo da dodate TRAP svom programu i to najbolje na samom početku

DEVICE NOT PRESENT ERROR (greška broj 5) Poruku o ovoj grešci dobijamo kada se obratimo nekoj periferiji — na primer štampaču disku ploteru U slučaju da ta periferija ne postoji da nije uključena ili da nije dobro priključena — dobićemo ovu poruku.

DIRECT MODE ONLY ERROR (greška broj 34) Kao što je poznato neke naredbe Basic-a kao AUTO, mogu se koristiti jedino u direktnom modu Ovakve naredbe ne mogu biti deo programa U slučaju da smo neku isključivo direktnu naredbu ubacili u program, dobićemo poruku o ovoj grešci Pritiskom na taster HELP dobićemo programsku liniju i naredbu koju smo greškom uneli u program

DIVISION BY ZERO ERROR (greška broj 20) Pri korišćenju matematičkih operacija moramo se pridržavati određenih pravila. U slučaju da neki broj pokušamo da podelimo sa nulom, a što je matematički nedopustivo dobićemo poruku o ovoj grešci

EXRTA IGNORED ERROR (greška bez broja) Do ove greške dolazi primenom INPUT naredbe U slučaju da u okviru podatka koji unosimo nakon INPUT naredbe upišemo i zarez (,), doći će do greške i kompjuter će nam ispisati ovu poruku

FILE DATA ERROR (greška broj 24) Pri korišćenju naredbi INPUT# ili GET#, da bismo uzeli podatke sa diska ili kasetofona dolazi do ove greške u slučaju da tražimo numerički a dobijemo alfa numerički podatak Da bismo otklonili ovu grešku, potrebno je da promenljivu koja se nalazi iza INPUT# ili GET# deklarišemo kao alfanumeričku (pridodamo joj znak za dolar)

FILE NOT FOUND ERROR (greška broj 4). Pri učitavanju programa sa diska ili kasetofona, ako ime programa napišemo pogrešno ili takav program ne postoji — dobićemo ovu poruku. Ako imate disk otkucajte prvo DIRECTORY kako biste dobili spisak svih programa na disku, i onda izvršite korekciju imena vašeg programa koji želite da učitate.

FILE NOT OPEN ERROR (greška broj 3). Pri korišćenju naredbi INPUT#br, GET#br ili Print#br moramo prethodno upotrebiti naredbu OPEN br,periferija. U slučaju da nismo upotrebili naredbu OPEN ili da smo napisali drugi broj kod OPEN naredbe nego kod naredbi INPUT#, GET# ili PRINT# — doći će do greške i dobićemo ovu poruku.

FILE OPEN ERROR (greška broj 2). Ako pokušamo da otvorimo datoteku koja je već otvorena, dobićemo poruku o ovoj grešci. Ako, pak, pokušamo da zatvorimo datoteku koja je već zatvorena, kompjuter neće javiti nikakvu grešku. Zato uvek pre pisanja OPEN naredbe, slobodno možemo napisati naredbu CLOSE. Na ovaj način smo sigurno izbegli ovu grešku.

FILE READ ERROR (greška broj 4i). Za vreme čitanja podataka sa diska došlo je do greške i dobićemo ovu poruku. Može se desiti da smo vrata na disku otvorili za vreme čitanja podataka.

FORMULA TOO COMPLEX ERROR (greška broj 25). Vrlo retka greška koja nastaje kada vršimo neko izučavanje sa previše zagrada. Zbog toga — ili pojednostavite matematički izraz (napišite ga u više programskih linija) ili obrišite nepotrebne zagrade.

ILLEGAL DEVICE NUMBER ERROR (greška broj 9). Neki periferni uređaji služe samo za izlaz, a neki samo za ulaz podataka. Na disk (periferija broj 8) možemo snimiti program, ali na ekran (periferija broj 3) ili štampač (periferija broj 4) ne možemo snimiti program. Do ove greške je došlo pogrešnim upisom drugog broja kod OPEN naredbe.

ILLEGAL DIRECT ERROR (greška broj 21). Ima naredbi Basic-a koje mogu da se koriste jedino u programskom, a ne u direktnom modu. Takva su, na primer, naredbe PRINT# ili INPUT#. Ako ovakve i slične naredbe pokušamo da koristimo u direktnom modu — dobićemo ovu poruku o grešci.

ILLEGAL QUANTITY ERROR (greška broj 14). Neke numeričke funkcije imaju ograničenje pri upotrebi brojeva. Tako, na primer, logaritam od nule ne postoji. U slučaju da tako nešto pokušamo da izračunamo dobićemo poruku o ovoj grešci. Najčešće ćemo činiti ovu grešku pri korišćenju funkcija READ i POKE. Drugi broj iza naredbe POKE mora biti u intervalu od 0 do 255. U slučaju da ovo nismo poštovali — doći će do greške.

LINE NUMBER TDO LARGE ERROR (greška broj 38). Programske linije moraju biti u intervalu od 0 do 63999. Ako koristimo funkciju RENUMBER, a pri tom dolazi do prekoračenja ovog intervala, dobićemo poruku o grešci. Pokušajte da renumerišete program sa manjim korakom.

LOAD ERROR (greška broj 29). Pri učitavanju programa sa diska ili kasetofona može doći do ove greške. Ova greška je mnogo češća pri učitavanju programa sa kasetofona nego sa diska. Ako je program snimljen na drugom kasetofonu, sa različitim uglom glave od vašeg — skoro je neminovno da će doći do ove greške.

LOOP NOT FOUND ERROR (greška broj 32). Svako DO koje započinje petlju zahteva odgovarajuće LOOP koje označava kraj. Kada WHILE naredba postane netačna, UNTIL istinita ili EXIT izvršena — program pokušava da pronađe LOOP naredbu koja označava kraj DO-LOOP. U slučaju da imamo naredbu DO a nemamo LOOP — doći će do ove greške.

LOOP WIDTHOUT DO ERROR (greška broj 33). Kada program naiđe na naredbu LOOP, onda će kompjuter potražiti naredbu DO koja je inicijalizovala petlju. U slučaju da ne nađe naredbu DO — dobićemo poruku o ovoj grešci.

MISSING FILE NAME ERROR (greška broj 8). Pri slanju podataka na neku periferiju blok podataka koji šaljemo ne mora posedovati ime. To je u slučaju slanja podataka na printer ili kasetofon. U slučaju da podatke šaljemo na disk, moramo navesti ime, a ako to nismo učinili doći će do ove greške. Proverite naredbe OPEN i DOPEN.

NEXT WITHOUT FOR ERROR (greška broj 10). Petlja mora početi naredbom FOR ... TO ... STEP, a mora se završiti sa NEXT. U slučaju da je program započeo petlju i došao do kraja a nije naišao na NEXT naredbu — javiće grešku. Proverite da li ste sve petlje koje ste započeli i završili.

NO GRAPHICS AREA ERROR (greška broj 35). Naredba GRAPHIC sa koje slede brojevi od 1 do 4 uzima 9 kilobajta memorijskog prostora za ekran visoke rezolucije. Da bismo koristili grafičke funkcije, moramo unapred rezervisati taj deo memorije funkcijom GRAPHIC. Ako to nismo učinili pa zadamo neku grafičku funkciju (na primer CIRCLE, DRAW, BOX ...) javiće se ova greška.

NOT INPUT FILE ERROR (greška broj 6). Pri otvaranju sekvencijalne ili programske datoteke određujemo da li vršimo čitanje ili upis podataka. Ako je datoteka otvorena za upis iz nje ne možemo čitati podatke. Ako pak to pokušamo da uradimo — dobićemo ovu poruku o grešci.

NOT OUTPUT FILE ERROR (greška broj 7). Pri otvaranju sekvencijalne ili programske datoteke određujemo da li vršimo čitanje ili upis podataka. Ako je datoteka otvorena za čitanje podataka sa neke periferije, tada je ne možemo koristiti za upis. Ako pak to pokušamo — dobićemo poruku o grešci.

OUT OF DATA ERROR (greška broj 13). Do ove greške dolazi pri korišćenju funkcija READ i DATA. Funkcijom READ čitamo podatke iz datoteke. Podataka u datoteci mora biti više ili jednako broju READ funkcija (ako je READ funkcija u nekoj petlji, računamo je onoliko puta koliko prolazi kroz tu petlju). U slučaju da podataka ima manje — doći će do poruke o ovoj grešci.

OUT OF MEMORY ERROR (greška broj 16). Postoje tri moguća uzroka ove greške. Prvo, ako je program u banci nula suviše veliki. Drugo, ako su promenljive popunile raspoloživu memoriju u banci jedan, i, treće, ako smo koristili suviše FOR-NEXT petlji (jednu u drugoj). Takođe ako koristimo previše funkcija GOSUB doći će do ove greške. Otkucajmo PRINT FRE(0),FRE(1) i dobićemo poruku o slobodnoj memoriji za program i ostale promenljive. Ako, pak, postoji dovoljno memorije a dobili smo ovu grešku — onda je stek-memorija prepunjena zbog previše otvorenih GOSUB funkcija.

OVERFLOW ERROR (greška broj 15). Najveći broj sa kojim Commodor 128 radi je približno 10 na 38 (jedan i 38 nula). Ako pri nekom izračunavanju dođe do pojave broja koji je veći od ovog — doći će do poruke o grešci.

REDIM'D ARRAY ERROR (greška broj 19). Kada koristimo neki niz promenljivih a indeks niza je veći od 10, neophodno je da taj niz promenljivih dimenzionišemo primenom naredbe DIM. Niz možemo samo jedanput dimenzionisati. Ako pokušamo da isti niz dva puta dimenzionišemo, dobićemo ovu poruku o grešci. Ako baš imamo potrebu da neki niz dva puta dimenzionišemo, prethodno moramo primeniti naredbu CLR.

REDO FROM START ERROR (greška bez broja). Kada u programu koristimo naredbu INPUT za numeričku promenljivu i kompjuter traži da ukucamo vrednost a mi umesto numeričke promenljive kucamo alfanumerički podatak — dobićemo ovu poruku o grešci. Pri ovoj grešci program se ne zaustavlja, već daje poruku o grešci i ponovo očekuje numeričku vrednost.

RETURN WITHOUT GOSUB ERROR (greška broj 12). Pri pisanju programa često ćemo koristiti potprograme. Odlazak na potprogram vršimo naredbom GOSUB, a povratak sa programa naredbom RETURN. Kad program naiđe na naredbu GOSUB, kompjuter programsku liniju zabeleži na stek-memoriju kako bi znao gde da se vrati nakon naredbe RETURN. Ako nismo stavili naredbu GOSUB a kompjuter je naišao na naredbu RETURN — dobićemo ovu poruku o grešci jer je stek-memorija prazna i kompjuter ne zna gde da se vrati u program.

STRING TOO LONG ERROR (greška broj 23). Kada radimo sa alfanumeričkim promenljivim moramo imati u vidu da one primaju maksimalno 255 karaktera. Ako ovaj broj prekoračimo, dobićemo ovu poruku o grešci. Ako nam je baš potreban niz karaktera koji je duži od 255, raščlanimo ga na više nizova karaktera tako da svaki pojedinačno ne bude duži od 255.

SYNTAX ERROR (greška broj 11). Ovu poruku ćemo najčešće viđati na ekranu. Prouzrokovana je pogrešno napisanom naredbom. Ako na primer, umesto PRINT napišemo PIRINT, dobićemo poruku o ovoj grešci. Pritiskom na taster HELP na ekranu će se pojaviti programska linija u kojoj je načinjena greška. U slučaju da radimo sa RGB monitorom, naredba koja je pogrešno napisana biće podvučena.

TOO MANY FILES ERROR (greška broj 1). Kod Commodora 128 možemo imati najviše 10 otvorenih fajli. Kada pokušamo da otvorimo 11, a pre toga nismo nijednu zatvorili — dobićemo ovu poruku o grešci. Ako je potrebno da otvorimo novu fajlu, moramo neku prethodno zatvoriti. Ovo je veoma retka greška, jer 10 fajli otvorenih u isto vreme je više nego dovoljno.

TYPE MISMATCH ERROR (greška broj 22). Ako nekoj alfanumeričkoj promenljivoj pokušamo da dodelimo numeričku vrednost, dobićemo ovu poruku o grešci. Na primer A$=1, kompjuter ne može da prihvati, a takođe ne može primeniti ni matematičke operacije na alfanumeričke podatke. Ako pokušamo da izračunamo kvadratni koren iz reči "KOMPJUTER" — takođe ćemo dobiti ovu poruku o grešci.

UNDEF'D FUNCTION ERROR (greška broj 27). Kada koristimo funkciju FN, kompjuter će potražiti naredbu DEF FN da bi pronašao definiciju funkcije. U slučaju da naredbu DEF FN ne pronađe doći će do greške. Da bismo koristili neku funkciju — moramo je prethodno definisati.

UNDEF'D STSTEMENT ERROR (greška broj 17). Pri korišćenju naredbi GOTO, GOSUB ili sličnih koje se odnose na brojeve programskih linija — može doći do greške ako broj programske linije koji je iza naredbe ne postoji. Vodite računa pri korišćenju funkcije COLLISION, jer naredba RENUMBER ne renumeriše broj programske linije iza te funkcije.

UNIMPLEMENTED COMMAND ERROR (greška broj 40). Postoje naredbe koje su bile planirane, ali nisu dodate Basic-u V 7.0 (na primer QUIT ili OFF). Ako njih primenimo, dobićemo ovu poruku o grešci. Za naredbe koje nisu ni planirane — dobićemo poruku SYNTAX ERROR.

UNRESOLVED REFERENCE ERROR (greška broj 39). Pri korišćenju funkcije RENUMBER dolazi do promene broja iza naredbi kao

što su **GOTO** ili **GOSUB**. Ako, pak, ove naredbe ukazuju na programsku liniju koja ne postoji u našem programu — doći će do poruke o ovoj grešci.

VERIFY ERROR (greška broj 28). Kada snimimo neki program na disk ili kasetofon, a zanima nas da li je on korektno snimljen, koristićemo naredbu VERIFY. U slučaju da program na disku ili kaseti nije potpuno identičan programu koji se nalazi u kompjuteru — dobićemo ovu poruku o grešci.

DODATAK „B"

KERNAL

Pod KERNAL-om podrazumevamo operativni sistem računara COMMODORE 128. To je mašinski program koji ima zadatak da kontroliše sve ulazno-izlazne funkcije, štampanje karaktera na ekranu, učitavanje i snimanje programa ... U memoriji kompjutera nalazi se od adrese $E000 do $FFFF. Deo koji ćemo u ovom poglavlju obraditi je njegova džamp-tabela. Pri pisanju mašinskih programa možemo koristiti određene rutine i potprograme iz operativnog sistema. Ako, na primer, želimo da učitamo neki program sa diska, rutine za učitavanje možemo uraditi u mašinskom jeziku. To će nam, međutim, oduzeti puno vremena i nepotrebno ćemo trošiti memoriju. Korišćenjem KERNAL-a dovoljno je da pozovemo određeni potprogram koji vrši učitavanje sa diska.

Operativni sistemi računara COMMODORE 64 i COMMODORE 128 imaju dosta i sličnosti i razlika. Pojedini blokovi memorije su potpuno isti, ali se oni nalaze na različitim mestima. To znači da mašinski programi koje smo pisali za C-64,a koji koriste određene rutine iz operativnog sistema, neće raditi na C-128. Prvenstveno zbog ovog razloga uvedena je džamp-tabela (tabela potprograma), tako da pri pisanju mašinskih programa, ako koristimo određene rutine iz operativnog sistema, treba da ih koristimo preko tabele potprograma. Takvi mašinski programi koji su pisani za COMMODORE 64 radiće i na računaru COMMODORE 128.

Firma COMMODORE kod svojih modela kompjutera menja operativni sistem, ali džamp-tabelu ne menja. To omogućuje da programi pisani na jednom kompjuteru bez problema rade na drugom.

Sada ćemo dati imena potprograma operativnog sistema, njegove pozivne adrese iz operativnog sistema i objašnjenja šta koji potprogram radi. Pošto postoje određeni potprogrami koje operativni sistem C-64 nema, iza imena potprograma u zagradi ćemo napisati za koji su kompjuter oni namenjeni. Džamp-tabelu dajemo uređenu po abecednom redu imena potprograma.

ACPTR (C-64 i C 128)

Svrha: Uzimanje bajta iz serijskog kanala
Pozivna adresa: $FFA5 (heksadekadna) 65445 (dekadna)

Korišćeni registri: A
Pripremni potprogrami: TALK, TKSA
Moguće greške: Koristi READSS

Opis funkcije: Ovaj potprogram ćemo koristiti kada želimo da dobijemo informacije od nekog perifernog uređaja koji je priključen na serijskom portu. Pozivanjem ovog potprograma dobijamo jedan bajt podataka sa serijskog kanala, i to preko akumulatora. Pre pozivanja ovog potprograma neophodno je da prethodno pozovemo potprogram TALK kako bi periferni uređaj koji je na serijskom kanalu poslao podatak. Ako periferni uređaj zahteva sekundarnu adresu, ona se mora poslati korišćenjem potprograma TKSA. Potprogram READST se koristi za očitavanje greške.

Kako se koristi:

(1) Korišćenjem potprograma TALK i TKSA kažemo perifernom uređaju da pošalje bajt
(2) Pozivamo potprogram ACPTR
(3) Preko akumulatora smo dobili poslati bajt

BASIN (CHRIN) (C 64 i C-128)

Svrha: Uzimanje bajta sa ulaznog kanala
Pozivna adresa: $FFCF (heksadekadna), 65487 (dekadna)
Korišćeni registri: A
Pripremni potprogrami: (CHKIN)
Moguće greške: Ako je u .A Carry flag setovan ili koristi READSS

Opis funkcije: Uzima bajt iz korišćene periferije i stavlja ga u akumulator. Ako prethodno nismo koristili nikakve pripremne potprograme, uzima se bajt sa tastature. Ako, pak, želimo da uzmemo bajt sa neke druge periferije, moramo koristiti potprogram CHKIN kako bismo definisali periferiju.

Kako se koristi:
(1) Preko potprograma CHKIN definišemo periferni uređaj
(2) Pozovemo potprogram BASIN
(3) Preko akumulatora smo dobili poslati bajt

BOOT CALL (C-128)

Svrha: Punjenje programa sa diska
Pozivna adresa: $FF53 (heksadekadna), 65363 (dekadna)
Korišćeni registri: A, X, Y
Pripremni potprogrami: nema
Moguće greške: I/O greška ako je Carry flag setovan

Opis funkcije: Potprogram služi za učitavanje programa sa diska. Program koji se na ovaj način učitava mora biti na disku kao Auto-Boot program. U slučaju da na disketi koja se nalazi u drajvu ne postoji Auto-Boot program — do učitavanja neće doći.

Kako se koristi:

(1) U akumulator postaviti ASCII vrednost broja diska.
(2) U X-registar postaviti broj periferije (0—31).
(3) Pozvamo potprogram BOOT CALL.

BSOUT (CHROUT) (C-64 i C-128)

Svrha: Slanje bajta preko kanala za izlaz
Pozivna adresa: $FFD2 (heksadekadna), 65490 (dekadna)
Korišćeni registri: .A
Pripremni potprogrami: (CKOUT)
Moguće greške: I/O greška ako je Carry flag setovan ili koristiti READSS

Opis funkcije: Potprogram nam koristi za slanje bajta kroz kanal koji je trenutno otvoren. Ako prethodno nismo definisali kanal za slanje, bajtovi koje šaljemo ovim potprogramom biće pokazani na ekranu. U slučaju kada želimo da bajtove šaljemo nekoj periferiji (na primer disku ili printeru), potrebno je prvo da pozovemo potprogram CHOUT kako bismo definisali periferiju kojoj šaljemo bajte.

Kako se koristi:

(1) Pozovemo potprogram CHOUT kako bismo definisali kanal za slanje podataka.
(2) U akumulator postavimo bajt koji želimo da pošaljemo.
(3) Pozovemo potprogram BSOUT.

CHKIN (C-64 i C-128)
Svrha: Postavljanje kanala za prijem podataka
Pozivna adresa: $FFC6 (heksadekadna), 65478 (dekadna)
Korišćeni registri: .A, .X, .Y
Pripremni potprogrami: OPEN
Moguće greške: I/O greška ako je Carry flag setovan ili koristiti READSS

Opis funkcije: Uloga ovog potprograma je postavljanje kanala za prijem podataka. Pre nego što pozovemo ovaj potprogram, prethodno moramo pozvati OPEN. Inače, ovaj potprogram se obavezno mora koristiti pre korišćenja potprograma BASIN i GETIN.

Kako se koristi:

(1) Pozovemo potprogram OPEN.
(2) U X-registar postavimo broj kanala koji smo definisali pri korišćenju potprograma OPEN.
(3) Pozovemo potprogram CHKIN.

CINT (C-64 i C-128)

Svrha: Inicijalizacija ekranskog editora
Poziyna adresa: $FF81 (heksadekadna), 65409 (dekadna)
Korišćeni registri: A, X, Y
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije Ovaj potprogram postavlja video-čip C-128 u normalno radno stanje. KERNAL ekranski editor je takođe inicijalizovan. Pre pozivanja ovog potprograma moramo postaviti mašinsku funkciju SEI. Pored toga, šesti bit na lokaciji $0A04 jednak je nuli. Ovaj potprogram će izvršiti kompletnu inicijalizaciju 40-kolonskog i 80-kolonskog ekrana, tastature, karakter ROM-a i registra za ton.

Kako se koristi:

(1) Pozvati potprogram CINT

CIOUT (C-64 i C-128)

Svrha: Slanje bajta preko serijskog kanala
Pozivna adresa: $FFAB(heksadekadna), 65448 (dekadna)
Korišćeni registri: A
Pripremni potprogrami: LISTN SECND
Moguće greške: Koristiti READSS

Opis funkcije Ova funkcija nam koristi za slanje podataka preko serijskog kanala. Da bismo korektno koristili ovaj potprogram, prethodno moramo definisati periferni uređaj kome šaljemo podatke. Određivanje perifernog uređaja vršimo potprogramom LISTN. Ako periferni uređaj zahteva sekundarnu adresu, nju moramo definisati koristeći funkciju SECND. Bajt koji želimo da pošaljemo moramo staviti u akumulator i pozvati ovu funkciju.

Kako se koristi:

(1) Pozvati potprogram LISTN
(2) Pozvati potprogram SECND
(3) U akumulator postaviti bajt koji šaljemo.
(4) Pozvati funkciju CIOUT

CKOUT (CHKOUT) (C-64 i C-128)

Svrha: Postavljanje kanala za slanje podataka
Pozivna adresa: $FFC9 (heksadekadna), 65481 (dekadna)
Korišćeni registri: A, X, Y
Pripremni potprogrami: OPEN
Moguće greške: Ako je u A Carry flag setovan ili koristi READSS

Opis funkcije. Ovo je među-potprogram. Koristi se za postavljanje kanala za slanje podataka. Pre poziva ove funkcije neophodno je da pre toga pozovemo potprogram OPEN. Inače, ovaj potprogram se koristi kada želimo da šaljemo podatke tako da pri korišćenju funkcije BSOUT prethodno moramo pozvati ovaj program. Pre nego što pozovemo CKOUT, u X-registar moramo postaviti broj kanala koji smo definisali u potprogramu OPEN.

Kako se koristi

(1) Pozovemo potprogram OPEN.
(2) U X-registar postavimo broj kanala
(3) Pozovemo potprogram CKOUT

CLALL (C-64 i C-128)

Svrha: Zatvaranje svih kanala
Pozivna adresa: $FFE7 (heksadekadna), 65511 (dekadna)
Korišćeni registri: A, X
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Ovaj potprogram zatvara sve kanale. Pored toga, on resetuje tabelu indeksa na lokaciji $0098. Nije mu potreban nijedan potprogram da bi korektno radio. Takođe, pre pozivanja uopšte nisu bitne vrednosti u registrima A, X ili Y.

Kako se koristi

(1) Pozvati potprogram CLALL

CLOSE (C-64 i C-128)

Svrha: Zatvaranje svih dosjea
Pozivna adresa: $FFC3 (heksadekadna), 65475 (dekadna)
Korišćeni registri: A, X, Y
Pripremni potprogrami: nema
Moguće greške: Ako je u A Carry flag setovan ili koristiti READSS

Opis funkcije. Ovaj potprogram se koristi za zatvaranje logičkih dosjea, pošto su sve I/O operacije završene. Pre njegovog pozivanja moramo akumulator napuniti brojem logičkog dosjea koji treba da se zatvori (isti broj koji je korišćen za otvaranje dosjea upotrebom OPEN funkcije). Treba napomenuti da se fajle koje su otvorene ka ekranu i tastaturi ne mogu zatvoriti.

Kako se koristi

(1) Napuniti akumulator brojem logičke fajle koja se želi zatvoriti
(2) Pozvati potprogram CLOSE

11*

CLOSE ALL (C-128)

Svrha. Zatvara sve fajle ka periferiji
Pozivna adresa: $FF4A (heksadekadna), 65354 (dekadna)
Korišćeni registri: A, X, Y
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Funkcija nam služi kako bismo zatvorili sve fajle ka određenoj periferiji. Pre pozivanja ovog potprograma potrebno je da akumulatoru dodelimo broj periferije (0—31) i zatim pozovemo ovaj potprogram.

Kako se koristi:

(1) Napuniti akumulator brojem periferije prema kojoj želimo da zatvorimo sve fajle.
(2) Pozvati potprogram CLOSE ALL.

CLRCH (CLRCHN) (C-64 i C-128)

Svrha. Oslobađa I/O kanale
Pozivna adresa: $FFCC (heksadekadna), 65484 (dekadna)
Korišćeni registri: A, X
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Ovaj potprogram pozivamo kako bismo obrisali sve otvorene kanale i postavili I/O kanale na njihove normalne vrednosti. Nakon njegovog pozivanja ulazni uređaj je postavljen na tastaturu, a izlazni na ekran. Napominjemo da funkcija CLALL automatski poziva ovaj potprogram.

Kako se koristi:

(1) Pozvati potprogram CLCHR.

C64MODE (C-128)

Svrha. Ulazak u mod 64
Pozivna adresa: $FF4D (heksadekadna), 65357 (dekadna)
Korišćeni registri: nema
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Postoji više načina da pređemo u mod 64. Iz mašinskog jezika ovo je najjednostavnije: potrebno je samo da pozovemo ovaj potprogram. U slučaju da nakon prelaska u mod 64 želimo da se

neki program startuje neophodno je da na adresama $8000 i $8001 postavimo niži i viši bajt startne adrese programa, a bajtovi od $8004 do $8009 moraju sadržati CBM80'

Kako se koristi
(1) Pozvati potprogram C64MODE.

DLCHR (C-128)

Svrha Inicijalizuje karakter set 80-kolonskog ekrana
Pozivna adresa $FF62 (heksadekadna), 65360 (dekadna)
Korišćeni registri .A, X
Pripremni potprogrami nema
Moguće greške nema

Opis funkcije Kada pozovemo ovaj potprogram doći će do kopiranja dela ROM-memorije (od adrese $D000 do $DFFF), gde se nalazi karakter ROM u deo memorije od adrese $2000 do $3FFF, i to u banci 14 Na taj način možemo koristiti 80-karakterni ekran. Pri inicijalizaciji sistema nakon uključivanja kompjutera ovaj potprogram se automatski poziva ako koristimo sliku na RGB monitoru Tako, ako smo koristili neki drugi karakter — set, originalni jednostavno možemo povratiti pozivanjem ove funkcije

Kako se koristi

(1) Pozvati potprograme DLCHR

DMA CALL (C-128)

Svrha. Slanje komandi za direktan pristup memoriji
Pozivna adresa $FF50 (heksadekadna) 65360 (dekadna)
Korišćeni registri .A X
Pripremni potprogrami nema
Moguće greške nema

Opis funkcije Ovu funkciju koristimo kada imamo proširenja RAM-memorije preko ketridža Služi nam za slanje komandi i rad sa dodatnom RAM-memorijom Možemo koristiti četiri komande i to 0-(STASH) kopiranje bloka memorije u dodatni RAM, 1-(FETCH) prebacivanje memorije iz dodatnog RAM-a u originalni, 2-(SWAP) izmena bloka memorije dodatnog RAM-a sa originalnim i 3-(VERIFY) provera da li je sadržaj bloka memorije u RAM-ketridžu isti sa blokom memorije u originalnom RAM-u

Kako se koristi

(1) Postavite niži i viši bajt adrese memorije za pristup na lokacije $DF02 i $DF03 redom
(2) Postaviti niži i viši bajt adrese memorije (proširenog RAM-a) na lokacije $DF04 i $DF05 redom

(3) Postaviti vrednost banke podataka proširenog RAM-a na lokaciju $DF06
(4) Postaviti niži i viši bajt broja podataka na adrese $DF07 i $DF08 redom.
(5) Postaviti u X-registar banku podataka originalne memorije.
(6) Postaviti u Y-registar broj komande (iz opisa funkcije 0—3).
(7) Pozvati potprogram DMA CALL

GETCFG (C-128)

Svrha. Uzimanje bajta za konfiguraciju MMU iz bilo koje banke
Pozivna adresa: $FF6B (heksadekadna) 65387 (dekadna)
Korišćeni registri .A
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije Ovaj potprogram služi za pravljenje konfiguracije memorije. Potrebno je da u X-registar postavimo broj banke (0—15) i pozovemo ovaj potprogram. Podatke o konfiguraciji MMU dobijamo pri povratku iz ovog potprograma preko akumulatora. Za izmenu banke podataka dovoljno je da stavimo naredbu STA $FF00 iza naredbe za pozivanje ovog potprograma.

Kako se koristi:

(1) U X-registar postavimo broj banke podataka (0—15)
(2) Pozvati potprogram GETCFG

GETIN (C-64 i C-128)

Svrha. Uzimanje bajta iz bafera za ulaz
Pozivna adresa: $FFE4 (heksadekadna) 65508 (dekadna)
Korišćeni registri .A,.X,.Y
Pripremni potprogrami: (CHKIN)
Moguće greške: ako je u a carry flag setovan ili koristi readss

Opis funkcije Ako je ulaz tastatura, ovaj potprogram uzima jedin simbol iz tastaturnog niza i vraća ga u akumulator kao ASCII vrednost. Ukoliko nismo pritisnuli nijedan taster, u akumulatoru će se naći vrednost nula. Kada kucamo na tastaturi bafer može da primi najviše 10 karaktera. Ako nijedan nismo uzeli, a nastavljamo da kucamo — kompjuter ce ignorisati buduće tastere koje pritiskamo. Inače ovaj potprogram automatski poziva funkciju BASIN za ulaz podataka preko serijskog kanala, a ako koristimo kasetofon — za unos podataka.

Kako se koristi

(1) Ako ne koristimo tastaturu kao ulaznu periferiju potrebno je da pozovemo potprogram CHKIN
(2) Pozvati potprogram GETIN

INDCMP (C 128)

Svrha: Poređenje podataka iz bilo koje banke
Pozivna adresa: $FF7A (heksadekadna), 65402 (dekadna)
Korišćeni registri: X
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije Potprogram koji nam omogućuje da upoređujemo brojeve ili blokove memorije između dve banke podataka. Njegova prednost je u tome što ne moramo prelaziti iz jedne u drugu banku podataka. Sve vreme se nalazimo u banci podataka u kojoj nam je i program i vršimo poređenja sa brojevima iz bilo koje banke.

Kako se koristi

(1) Potrebno je da niži i viši bajt početka bloka memorije koji želimo da poredimo — postavimo negde u nultoj strani (adrese od $00 do $FF).

(2) Vrednost adrese nižeg bajta moramo postaviti na lokaciju $02C8.

(3) Akumulator napunimo bajtom koji želimo da poredimo.

(4) Napunimo Y-indeks registar kao da koristimo funkciju CMP (baza),Y

(5) X registar napunimo brojem banke u kojoj se nalazi blok podataka koji želimo da poredimo.

(6) Pozovemo potprogram INDCMP

Napomena Ako vršimo poređenje u istoj banci podataka u kojoj se nalazi i program (nema promene banke), ovaj potprogram je identičan mašinskoj funkciji CMP (baza),Y

INDFET (C-128)

Svrha: Uzimanje bajta iz bilo koje banke podataka
Pozivna adresa: $FF74 (heksadekadna) 65396 (dekadna)
Korišćeni registri: A, X
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije Ovaj potprogram nam omogućuje da uzmemo bajt ili blok podataka iz bilo koje banke podataka. Prednost ove funkcije je u tome što za uzimanje bloka memorije iz bilo koje banke podataka ne moramo napuštati banku podataka u kojoj se nalazi naš program.

Kako se koristi

(1) Potrebno je da niži i viši bajt početka bloka memorije, čije bajtove želimo da koristimo, postavimo negde u nultoj strani (adrese od $00 do $FF).

(2) Vrednost adrese nižeg bajta moramo postaviti u akumulator

(3) Napunimo Y-indeks registar kao da koristimo funkciju LDA (baza),Y

(4) X-registar napunimo brojem banke u kojoj se nalazi blok podataka koji želimo da poredimo

(5) Pozovemo potprogram INDFET

Napomena. Ako koristimo podatke iz iste banke u kojoj se nalazi i naš program, ovaj potprogram je potpuno isti kao i mašinska funkcija LDA (baza),Y

INDSTA (C-128)

Svrha: Postavljanje bajta na bilo koju memorijsku lokaciju u bilo kojoj banci
Pozivna adresa $FF77 (heksadekadna), 65399 (dekadna)
Korišćeni registri .X
Pripremni potprogrami nema
Moguće greške nema

Opis funkcije Ovaj potprogram nam omogućuje da postavimo bajt ili blok podataka u bilo koju banku podataka. Prednost ove funkcije je u tome što za postavljanje bloka memorije u bilo koju banku podataka ne zahteva napuštanje banke u kojoj se nalazi naš program

Kako se koristi.

(1) Potrebno je da niži i viši bajt početka bloka memorije, gde postavljamo naše podatke — postavimo negde u nultoj strani (adrese od $00 do $FF)

(2) Vrednost adrese nižeg bajta moramo postaviti na lokaciju $02C8

(3) Akumulator napunimo bajtom koji želimo da postavimo

(4) Napunimo Y-indeks registar kao da koristimo funkciju STA (baza),Y

(5) X registar napunimo brojem banke u kojoj se nalazi blok podataka koji želimo da poredimo

(6) Pozovemo potprogram INDSTA.

Napomena: Ako postavljamo podatke u istu banku u kojoj se nalazi i naš program, ovaj potprogram je potpuno isti kao i mašinska funkcija STA (baza) Y

IOBASE (C-64 i C-128)

Svrha. Definisanje I/O memorijske strane
Pozivna adresa $FFF3 (heksadekadna), 65523 (dekadna)
Korišćeni registri X, Y
Pripremni potprogrami nema
Moguće greške nema

Opis funkcije. Ovim potprogramom se X i Y-registri postavljaju tako da određuju viši i niži bajt memorijskog dela gde se nalaze I/O registri. Ova adresa se zatim može koristiti nešto 'pomerena' — da bi se omogućio pristup memorijskom delu gde se nalaze I/O uređaji. Ovaj 'pomeraj' adrese predstavlja broj lokacija od početka memorijske strane na kojoj se nalazi željeni I/O registar. X-registar sadrži niže bajtove, dok Y-registar sadrži više bajtove adrese.

Kako se koristi.

(1) Pozvati ovaj potprogram (IOBASE)
(2) Stavite X i Y-registre u uzastopne lokacije
(3) Napunite Y-registar sa 'pomerenom' lokacijom adrese
(4) Pristupate toj I/O lokaciji

Napomena. Ovaj potprogram postoji da obezbedi usklađenost između COMMODORE-ovih modela kompjutera (C-64, VIC-20, C-128, ...). Ako dođe do izmene operativnog sistema ili položaja I/O lokacija, ovaj potprogram nam omogućuje da prevaziđemo te razlike.

IOINIT (C-64 i C-128)

Svrha: Inicijalizacija I/O periferija
Pozivna adresa: $FF84 (heksadekadna); 65412 (dekadna)
Korišćeni registri: A, X, Y
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Pre pozivanja ove rutine neophodno je da izvršimo mašinsku funkciju SEI, jer ova funkcija inicijalizuje I/O periferije. Ako je sedmi bit na lokaciji $0A04 jednak nuli, doći će do kompletne inicijalizacije. Pod tim podrazumevamo inicijalizaciju obe CIE, VIC-čipa i čipa koji kontroliše sliku na RGB monitoru. Ako je, pak, ovaj bit setovan, tj. ako je jednak jedinici, pri izvršenju ovog potprograma neće doći do inicijalizacije čipa koji kontroliše sliku na RGB monitoru.

Kako se koristi.

(1) Postaviti jedinicu ili nulu na sedmi bit registra $0A04
(2) Pozvati ovaj potprogram

JMPFAR (C-128)

Svrha: Skok na program koji se nalazi u bilo kojoj banci
Pozivna adresa: $FF71 (heksadekadna); 65393 (dekadna)
Korišćeni registri: nijedan
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije Ovaj potprogram nam služi kako bi naš mašinski program mogao da startuje drugi program koji se nalazi u bilo kojoj banci podataka. Ovo je veoma koristan potprogram, jer ne moramo da pređemo u neku banku podataka da bismo u njoj startovali neki mašinski program

Kako se koristi

(1) Postaviti broj banke u kojoj želimo da startujemo program na memorijsku lokaciju 2
(2) Postaviti viši i niži bajt startne adrese na lokacije 3 i 4
(3) Vrednost status registra postaviti na lokaciju 5
(4) Vrednosti akumulatora X-registra i Y registra postaviti respektivno na memorijske lokacije 6 7 i 8
(5) Pozvati potprogram JMPFAR

JSRFAR (C-128)

Svrha Skok na potprogram koji se nalazi u bilo kojoj banci
Pozivna adresa $FF6E (heksadekadna) 65390 (dekadna)
Korišćeni registri nijedan
Pripremni potprogrami nema
Moguće greške nema

Opis funkcije Ovaj potprogram nam služi kako bi naš mašinski program mogao da startuje potprogram koji se nalazi u bilo kojoj banci podataka. Ovo je veoma koristan potprogram, jer ne moramo da pređemo u neku banku podataka da bismo u njoj startovali neki mašinski potprogram

Kako se koristi

(1) Postaviti broj banke u kojoj želimo da startujemo program na memorijsku lokaciju 2
(2) Postaviti viši i niži bajt startne adrese na lokacije 3 i 4
(3) Vrednost status-registra postaviti na lokaciju 5
(4) Vrednosti akumulatora X-registra i Y-registra postaviti respektivno na memorijske lokacije 6 7 i 8
(5) Pozvati potprogram JSRFAR
(6) Za povratak u program tj za povratak u banku podataka gde se nalazi naš izvorni program napuniti status-registar akumulator, X-registar i Y registar vrednostima koje se nalaze respektivno na memorijskim lokacijama 5 6, 7 i 8

KEY (SCNKEY) (C-64 i C 128)

Svrha. Potprogram koji čita tastaturu
Pozivna adresa $FF9F (heksadekadna) 65439 (dekadna)
Korišćeni registri nijedan
Pripremni potprogrami nema
Moguće greške nema

Opis funkcije Ovaj potprogram pretražuje tastaturu i proverava da li je neki taster pritisnut. Ukoliko je neki taster pritisnut, njegova ASCII vrednost se smešta u bafer za tastaturu. Treba napomenuti da se ovaj potprogram poziva samo u slučaju ako je isključena normalna IRQ rutina.

Kako se koristi

(1) Pozvati potprogram KEY

LISTN (LISTEN) (C-64 i C-128)

Svrha: Dovodi serijski kanal u stanje prijema
Pozivna adresa: $FFB1 (heksadekadna) 65457 (dekadna)
Korišćeni registri: .A
Pripremni potprogrami: nema
Moguće greške: koristi READSS

Opis funkcije Potprogram dovodi serijski kanal u stanje prijema podataka. Akumulator mora biti napunjen odgovarajućim brojem uređaja iz opsega od 0 do 31 — pre nego što pozovemo ovaj potprogram. Potprogram će analizirati navedeni bajt, i bit po bit da bi prešao u adresu za prijem. Zatim će preneti ovaj bajt kao naredbu serijskom kanalu. Određeni uređaji će tada preći u prijemni mod i biti spremni da prime informacije.

Kako se koristi

(1) Napuniti akumulator brojem periferije kojoj se obraćamo (0—31)
(2) Pozvati ovaj potprogram

LKUPLA (C-128)

Svrha: Pronalaženje broja fajle u tabeli za fajle
Pozivna adresa: $FF59 (heksadekadna) 65369 (dekadna)
Korišćeni registri: .A, .X, .Y
Pripremni potprogrami: nema
Moguće greške: Ako je Carry flag setovan, fajla nije pronađena

Opis funkcije Napuniti akumulator logičkim brojem fajle i pozvati ovaj potprogram. Ako je pronađena dobićemo sledeće informacije: broj logičke fajle će ostati u akumulatoru, broj periferije biće smešten u X-registru a broj sekundarne adrese nalaziće se u Y-registru.

Kako se koristi

(1) Napuniti akumulator logičkim brojem fajle
(2) Pozvati potprogram LKUPLA

LKUPSA (C 128)

Svrha Pronalaženje sekundarne adrese iz tabele za fajle ci
Pozivna adresa $FF5C (heksadekadna), 65372 (dekadna)
Korišćeni registri .A X, Y
Pripremni potprogrami nema
Moguće greške Ako je Carry flag setovan, fajla nija pronađena

Opis funkcije Napuniti Y-registar sekundarnom adresom i pozvati ovaj potprogram Ako je fajla pronađena dobićemo sledeće podatke broj logičke fajle biće u akumulatoru, broj periferije biće smešten u X-registru a broj sekundarne adrese ostaće u Y-registru:

Kako se koristi

(1) Napuniti Y-registar sekundarnom adresom
(2) Pozvati potprogram LKUPSA

LOAD (C-64 i C-128)

Svrha Učitavanje ili verifikacija podataka sa periferije
Pozivna adresa $FFD5 (heksadekadna), 65493 (dekadna)
Korišćeni registri .A, X Y
Pripremni potprogrami SETNAM, SETBNK i SETLFS
Moguće greške Ako je Carry flag setovan ili koristi READSS

Opis funkcije Potprogram koji nam služi da bismo učitali u RAM kompjutera podatke sa diska ili kasetofona Pre pozivanja ovog potprograma neophodno je da pozovemo sledeće tri funkcije SETNAM SETBNK i SETLFS U opisu svake od ovih funkcija piše koja je njena uloga Nakon toga napunimo akumulator sa 0 — u slučaju da želimo podatke da učitamo Ako akumulator napunimo nekom drugom vrednošću koja je različita od nule, kompjuter će izvršiti verifikaciju Ako je sekundarna adresa jednaka nuli napuniti X i Y-registar nižim odnosno višim bajtom početne adrese memorije gde želimo da se učita ili verifikuje program

Kako se koristi

(1) Pozvati potprograme SETNAM, SETBNK i SETLFS

(2) Napuniti akumulator sa 0 ili 1 — u zavisnosti od toga da li želite učitavanje ili verifikaciju

(3) Napuniti X i Y registre nižim i višim bajtom početne adrese memorijske lokacije od koje želimo da počne učitavanje ili verifikacija

(4) Pozvati potprogram LOAD

MEMBOT (C-64 i C 128)

Svrha: Postavljanje ili čitanje donje granice memorije
Pozivna adresa: $FF9C (heksadekadna), 65436 (dekadna)
Korišćeni registri: .X, Y
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Zbog postojanja banki podataka, ovaj potprogram nema neku praktičnu primenu u modu C-128. On u stvari postoji da bi se održala kompatibilnost Commodore-ovih kompjutera. Za postavljanje donje granice memorije ($0A05—$0A06), očistiti Carry flag, postaviti niži i viši bajt donje granice memorije u X odnosno Y-registar i pozvati ovu rutinu. Za čitanje, pak, potrebno je setovati Carry flag i pozvati ovaj potprogram. Vrednosti (niži i viši bajt) dobijamo preko X odnosno Y-registra.

Kako se koristi:

(1) Postaviti 0 ili 1 u Carry flag — u zavisnosti od toga da li želimo da postavimo ili pročitamo donju granicu memorije.

(2) Za postavljanje granice memorije postaviti niži i viši bajt u X, tj. Y-registar. Ako smo Carry flag postavili na jedan — ovaj korak se preskače.

(3) Pozvati potprogram MEMBOT.

MEMTOP (C-64 i C-128)

Svrha: Postavljanje ili čitanje granice memorije
Pozivna adresa: $FF99 (heksadekadna), 65433 (dekadna)
Korišćeni registri: .X, Y
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Zbog postojanja banki podataka ovaj potprogram nema neku praktičnu primenu u modu C-128. I on postoji da bi se održala kompatibilnost Commodore-ovih kompjutera. Za postavljanja gornje granice memorije ($0A07—$0A08), očistiti Carry flag, postaviti niži i viši bajt donje granice memorije u X odnosno Y-registar i pozvati ovu rutinu. Za čitanje, pak, potrebno je setovati Carry flag i pozvati ovaj potprogram. Vrednosti (niži i viši bajt) dobijamo preko X odnosno Y-registra.

Kako se koristi:

(1) postaviti 0 ili 1 u Carry flag — u zavisnosti od toga da li želimo da postavimo ili pročitamo gornju granicu memorije.

(2) Za postavljanje granice memorije postaviti niži i viši bajt u X ili Y-registar. U slučaju da smo Carry flag postavili na jedan — ovaj korak se preskače

(3) Pozvati potprogram MEMTOP

OPEN (C-64 i C-128)

Svrha: Otvaranje logičke fajle
Pozivna adresa: $FFC0 (heksadekadna); 65472 (dekadna)
Korišćeni registri: A, X, Y
Pripremni potprogrami: SETNAM, SETBNK i SETLFS
Moguće greške: Ako je u A Carry flag setovan ili koristi READSS

Opis funkcije: Ovaj potprogram koristimo za otvaranje logičke fajle. Pre pozivanja ovog potprograma neophodno je prethodno pozvati sledeće funkcije: SETNAM, SETBNK i SETLFS

Kako se koristi:

(1) Pozovemo potprogram SETNAM, SETBNK i SETLFS
(2) Pozovemo potprogram OPEN

PFKEY (C-128)

Svrha: Programiranje funkcijskih tastera
Pozivna adresa: $FF65 (heksadekadna); 65381 (dekadna)
Korišćeni registri: A, X, Y
Pripremni potprogrami: nema
Moguće greške: ako je u Carry flag setovan

Opis funkcije: Ovaj potprogram nam služi za definisanje funkcijskih tastera. Potrebno je da naziv naredbe (u ASCII kodu) postavimo u RAM-memoriju. Zatim adresu naziva (niži i viši bajt kao i banku podataka) postavimo negde u RAM-memoriji u nultoj strani. U akumulator postavimo adresu prvog bajta iz nulte strane, gde smo postavili prethodne vrednosti, a u X-registar učitamo vrednost od 0 do 10 — u zavisnosti od toga koji funkcijski taster želimo da definišemo. U Y-registar postavimo dužinu naziva funkcije i pozovemo ovu rutinu.

Kako se koristi:

(1) Postavimo niži, viši bajt i broj banke podataka gde se nalazi naziv funkcije u tri bajta na nultoj strani
(2) Adresu prvog bajta iz nulte strane, gde smo postavili pret hodna tri parametra, postavimo u akumulator
(3) U X-registar učitamo vrednost funkcijskog tastera (0—10)
(4) U Y-registar postavimo dužinu naziva funkcije
(5) Pozovemo potprogram PFKEY

PHOENIX (C-128)

Svrha: Inicijalizacija funkcijskog ROM ketridža.
Pozivna adresa: $FF56 (heksadekadna), 65366 (dekadna)
Korišćeni registri: .A, X, Y
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Ovaj potprogram se poziva kada priključimo funkcijski ROM ketridž radi njegove inicijalizacije.

Kako se koristi

(1) Pozvati potprogram PHOENIX

PLOT (C-64 i C-128)

Svrha: Postavljanje ili čitanje pozicije kursora
Pozivna adresa: $FFF0 (heksadekadna) 65520 (dekadna)
Korišćeni registri: .X .Y
Pripremni potprogrami: nema
Moguće greške: Ako je Carry flag setovan, pozicija je van dela ekrana definisanog WINDOW funkcijom

Opis funkcije. Za postavljanje pozicije kursora neophodno je očistiti Carry flag. U X i Y-registar postaviti poziciju kursora i pozvati ovu rutinu. Za čitanje pozicije kursora Carry flag mora biti setovan a nakon pozivanja ove rutine poziciju kursora dobijamo preko X i Y-registara. Poziciju dobijamo relativno u odnosu na korišćenu funkciju WINDOW.

Kako se koristi

(1) Za postavljanje pozicije kursora očistiti Carry flag. Za čitanje on mora biti setovan.
(2) U X i Y registar postaviti poziciju kursora. Ako je Carry flag jedan, ovaj korak se preskače.
(3) Pozvati potprogram PLOT

PRIMM (C-128)

Svrha: štampanje niza ASCII karaktera
Pozivna adresa: $FF7D (heksadekadna) 65405 (dekadna)
Korišćeni registri: nema
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Kada želimo neki tekst da prikažemo na ekranu, za to možemo koristiti mnogo postupaka. Ovaj potprogram nam pruža mogućnost da to uradimo na najlakši način. Ispred bloka AASCII ka

raktera koje želimo da odštampamo — potrebno je da pozovemo ovaj potprogram (obavezno sa JSR, a ne sa JMP), a na kraju bloka poda taka, čija dužina ne sme biti veća od 255 bajta, stavimo 0

Kako se koristi.

(1) Pozovemo potprogram PRIMM na koga sledi niz ASCII kodova na čijem kraju se nalazi nula

RAMTAS (C-64 i C-128)

Svrha: Inicijalizacija RAM memorije i bafera
Poziva adresa: $FF87 (heksadekadna), 65415 (dekadna)
Korišćeni registri: .A, .X, .Y
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Ovaj potprogram briše kompletnu nultu stranu, vrši povraćaj, briše bafer i inicijalizuje podnožje i vrh RAM-memorije. On takođe postavlja sistemski vektor ($0A00—$0A01) na hladni start Basic-a ($4000).

Kako se koristi.

(1) Pozvati potprogram RAMTAS

RDTIM (C-64 i C-128)

Svrha: Čitanje vrednosti časovnika sistema
Poziva adresa: $FFDE (heksadekadna), 65502 (dekadna)
Korišćeni registri: .A, .X, .Y
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Ovaj potprogram nam služi da bismo pročitali vrednosti sistemskog CLOCK-a sistema. Nakon poziva ovog potprograma dobićemo nižu, srednju i višu vrednost sistemskog sata — redom u registrima A, X i Y

Kako se koristi.

(1) Pozvati potprogram RDTIM

READSS (READST) (C-64 i C-128)

Svrha: Čitanje I/O statusa
Poziva adresa: $FFB7 (heksadekadna), 65463 (dekadna)
Korišćeni registri: .A
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije: Ovaj potprogram nam saopštava koji je poslednji status I/O operacija. Poruku dobijamo preko akumulatora. U zavisnosti od toga koja je vrednost u akumulatoru — znamo stanje prikazano u sledećoj tabeli:

Akum.	Kasetofon	RS-232	Serijski port
1	—	greška PARITY	nije za zapis
2	—	greška sastava	nije za ulaz
4	kratak blok	bafer je pun	—
8	dugačak blok	bafer je prazan	—
16	greška čitanja	nedostaje CTS	greška VERIFY
32	pogrešna suma	—	—
64	kraj podataka	nedostaje DSR	kraj podataka
128	kraj trake	prekid prijema	periferija DFF

Kako se koristi:

(1) Pozvati potprogram READSS

RESTOR (C-64 i C-128)

Svrha: Povraćaj indirektnih KERNAL vektora
Pozivna adresa: $FF8A (heksadekadna), 65418 (dekadna)
Korišćeni registri: .A, .X, .Y
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije: Ovaj potprogram se koristi za uspostavljanje normalnih vrednosti KERNAL vektora, računajući i vektor interapta. Zbog eventualne promene vektora interapta, pre pozivanja ove funkcije neophodno je postaviti mašinsku instrukciju SEI.

Kako se koristi:

(1) Otkucati mašinsku instrukciju SEI
(2) Pozvati potprogram RESTOR

SAVE (C-64 i C-128)

Svrha: Snimanje sadržaja memorije na periferni uređaj
Pozivna adresa: $FFD8 (heksadekadna), 65496 (dekadna)
Korišćeni registri: .A, .X, .Y
Pripremni potprogrami: SETNAM, SETBNK i SETLFS
Moguće greške: Ako je u akumulatoru Carry flag setovan

Opis funkcije: Kada želimo da neki deo memorije snimamo na disk ili kasetofon — koristićemo ovaj potprogram. Prethodno je neophodno pozvati funkcije SETNAM, SETBNK i SETLFS. Nakon toga niži i viši bajt početne adrese bloka koji želimo da snimamo posta-

viti negde u nultoj strani. Akumulator napuniti adresom iz nulte strane gde smo postavili ova dva bajta. Niži i viši bajt krajnje adrese uvećati za jedan i vrednosti dodeliti X odnosno Y-registru. Nakon toga pozivamo ovaj potprogram.

Kako se koristi.

(1) Pozvati potprograme, SETNAM, SETBNK i SETLFS.
(2) Početnu adresu postaviti na nultu stranu.
(3) U akumulator postaviti vrednost adrese iz nulte strane — gde smo postavili prethodna dva bajta.
(4) U X i Y-registar postaviti niži i viši bajt krajnje adrese bloka memorije koji snimamo, i to uvećan za jedan.
(5) Pozvati potprograma SAVE.

SCRORG (SCREEN) (C-64 i C-128)

Svrha: Čitanje veličine ekrana
Pozivna adresa: $FFED (heksadekadna); 65517 (dekadna)
Korišćeni registri: .A, .X, .Y
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Ova funkcija nam koristi za uzimanje podataka o veličini ekrana koji trenutno koristimo. Nakon njenog pozivanja podatke dobijamo preko akumulatora. Vrednost širine ekrana nalazi se u X-registru, a vrednost visine ekrana u Y-registru.

Kako se koristi.

(1) Pozvati potprograme SCRORG.

SECND (SECOND) (C-64 i C-128)

Svrha: Postavljanje sekundarne adrese
Pozivna adresa: $FF93 (heksadekadna), 65427 (dekadna)
Korišćeni registri: .A
Pripremni potprogrami: LISTN
Moguće greške: Koristi READSS

Opis funkcije. Pre poziva ovog potprograma obavezno moramo pozvati funkciju LISTN. Inače, ovaj potprogram koristimo za postavljanje sekundarne adrese pri otvaranju fajle ka nekoj periferiji. Vrednost sekundarne adrese nalazi se u akumulatoru.

Kako se koristi.

(1) Pozvati potprogram LISTN.
(2) U akumulator postaviti vrednost sekundarne adrese.
(3) Pozvati potprogram SECND.

SETBNK (C-128)

Svrha: Postavljanje banke podataka za I/O operacije
Pozivna adresa: $FF68 (heksadekadna), 65384 (dekadna)
Korišćeni registri: nijedan
Pripremni potprogrami: SETNAM
Moguće greške: nema

Opis funkcije. Pri radu sa diskom ili kasetofonom, podatke možemo poslati, uzeti ili izvršiti proveru ispravnosti snimljenih podataka. Sa kojom bankom podataka ćemo raditi — to određujemo pomoću ove funkcije. Prethodno moramo pozvati potprogram SETNAM — kako bismo dali ime podacima, a zatim napuniti akumulator brojem banke u kojoj se nalazi blok podataka koji snimamo ili broj banke u koju hoćemo da učitamo neki program. U X-registar postavimo broj banke u kojoj se nalazi ime bloka podataka. Nakon toga možemo pozvati ovaj potprogram.

Kako se koristi:

(1) Pozvati potprogram SETNAM
(2) U akumulator postaviti broj banke sa kojom radimo
(3) U X-registar postaviti broj banke u kojoj se nalazi ime bloka podataka sa kojim radimo
(4) Pozvati potprogram SETBNK

SETLFS (C-64 i C-128)

Svrha: Definisanje broja fajle, broja periferije i sekundarne adrese
Pozivna adresa: $FFBA (heksadekadna), 65466 (dekadna)
Korišćeni registri: nijedan
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije. Ovaj potprogram je potpuno isti kao Basic-naredba OPEN A,B,C. U akumulator, X i Y-registar postavimo broj fajle, broj periferije i sekundarnu adresu. Nakon toga pozovemo ovaj potprogram. U slučaju da periferija ne zahteva sekundarnu adresu, u Y-registar postavimo vrednost 255.

Kako se koristi:

(1) U akumulator postavimo broj logičke fajle
(2) U X-registar postavimo broj periferije sa kojom ćemo vršiti komunikaciju
(3) U Y-registar postavimo sekundarnu adresu
(4) Pozvati potprogram SETLFS

12*

SETMSG (C-64 i C-128)

Svrha: Uključivanje ili isključivanje KERNAL poruka
Pozivna adresa: $FF90 (heksadekadna), 65424 (dekadna)
Korišćeni registri: nijedan
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije: Pri izvršavanju nekog mašinskog programa kompjuter nam povremeno daje određene poruke. To mogu biti poruke o greškama ili kontrolne poruke (na primer, poruka PRESS PLAY ON TAPE). Pošto se ovakve poruke ispisuju na ekranu, a nama je ekranska memorija već zauzeta, postavlja se zahtev da se ovakve poruke ne ispisuju. To opslužemo ovim potprogramom.

Kako se koristi

(1) Ako želimo da nam kompjuter ispisuje poruke o greškama, potrebno je da setujemo 6 bitova u akumulatoru
(2) Ako želimo da nam kompjuter ispisuje kontrolne poruke, potrebno je da setuje 7. bit u akumulatoru.
(3) Pozvati potprogram SETMSG

SETNAM (C-64 i C-128)

Svrha: Postavljanje imena fajle
Pozivna adresa: $FFBD (heksadekadna), 65469 (dekadna)
Korišćeni registri: nijedan
Pripremni potprogrami: SETBNK
Moguće greške: nema

Opis funkcije: Ovaj potprogram nam služi za dodeljivanje imena nekoj fajli. Pri radu sa diskom ili kasetofonom bilo da učitavamo ili snimamo neke podatke, potrebno je da damo ime tim podacima. Za definisanje mesta u memoriji kompjutera gde se nalazi ime fajle koristimo ovu funkciju.

Kako se koristi.

(1) Pozvati potprogram SETBNK
(2) Postaviti ime fajle ASCII karakterima negde u RAM-memoriju
(3) U akumulator učitati dužinu imena
(4) U X i Y registar učitati niži i viši bajt adrese imena fajle
(5) Ako ime fajle nije neophodno, u akumulator učitati nulu
(6) Pozvati potprogram SETNAM

SETTIM (C-64 i C-128)

Svrha: Postavljanje parametara časovnika sistema
Pozivna adresa: $FFDB (heksadekadna), 65499 (dekadna)
Korišćeni registri: nijedan
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije: Ovaj potprogram se koristi za postavljanje sva tri parametra sata kompjutera. U akumulator, X i Y-registar postavimo redom niži, srednji i viši bajt za podešavanje sata, pa pozovemo ovaj potprogram.

Kako se koristi:

(1) U akumulator postaviti niži bajt za podešavanje sata
(2) U X-registar postaviti srednji bajt za podešavanje sata
(3) U Y-registar postaviti viši bajt za podešavanje sata
(4) Pozvati potprogram SETTIM

SETTMO (C-64 i C-128)

Svrha: Postavljanje IEEE na pauzu
Pozivna adresa: $FFA2 (heksadekadna), 65442 (dekadna)
Korišćeni registri: nijedan
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije: Ovaj potprogram se koristi u modu 64 kada je priključen IEEE ketridž. Da bismo postavili komunikaciju na pauzu, potrebno je da pre pozivanja ovog potprograma postavimo nulu na sedmi bit vrednosti akumulatora.

Kako se koristi:

(1) Postaviti 0 ili 1 na sedmi bit u akumulatoru — u zavisnosti od toga da li želimo pauzu ili ne
(2) Pozvati potprogram SETTMO

SPIN SPOUT (C-128)

Svrha: Postavljanje serijskog porta na brzi režim rada
Pozivna adresa: $FF47 (heksadekadna), 65351 (dekadna)
Korišćeni registri: A
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije: Ovaj potprogram se koristi samo ako imamo disk-jedinicu 1571. Služi nam da bismo uspostavili bržu komunikaciju sa disk-jedinicom. Pri korišćenju naredbi LOAD ili SAVE, kompjuter automatski prelazi u brži režim komunikacije.

Kako se koristi:

(1) Postavimo nulu u Carry flag za brži režim rada
(2) Pozovimo ovaj potprogram

STOP (C-64 i C-128)

Svrha: Čitanje RUN/STOP tastera
Pozivna adresa: $FFE1 (heksadekadna), 65505 (dekadna)
Korišćeni registri: .A, .X
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije: Ako je taster RUN/STOP na tastaturi pritisnut, dolazi do automatskog poziva potprograma CLRCH, brisanja bafera za tastaturu i povratka u program sa setovima Z flagom.

Kako se koristi:

(1) Pozvati potprogram STOP

SWAPPER (C-128)

Svrha: Prebacivanje ekrana između modova 40 i 80 karaktera u redu
Pozivna adresa: $FF5F (heksadekadna), 65375 (dekadna)
Korišćeni registri: .A, .X, .Y
Pripremni potprogrami: nema
Moguće greške: nema

Opis funkcije: Commodore 128 ima dva ekrana za rad sa tekstom. Jedan dobijamo preko televizora ili kompozitnog monitora, a drugi dobijamo preko RGB monitora. Ovaj program nam služi kako bismo vršili prelazak iz jednog moda u drugi.

Kako se koristi:

(1) Za mod od 80 karaktera setovati sedmi bit na adresi $D7
(2) Pozvati potprogram SWAPPER

TALK (C-64 i C-128)

Svrha: Komanda serijskom uređaju da počne sa slanjem podataka
Pozivna sredstva: $FFB4 (heksadekadna), 65460 (dekadna)
Korišćeni registri: .A
Pripremni potprogrami: nema
Moguće greške: Koristi READSS

Opis funkcije: Ovaj potprogram se koristi — da bismo naredili serijskom uređaju da počne sa slanjem podataka. Pre pozivanja ovog potprograma neophodno je da akumulator uzme vrednost (0—31) perifernog uređaja sa kojim vršimo komunikaciju.

Kako se koristi:

(1) U akumulator postaviti broj perifernog uređaja (0—31)
(2) Pozvati potprogram TALK

TKSA (C-64 i C 128)

Svrha. Slanje sekundarne adrese uređaju kome je naređeno da predaje
Pozivna adresa $FF96 (heksadekadna) 65430 (dekadna)
Korišćeni registri .A
Pripremni potprogrami TALK
Moguće greške. Koristi READSS

Opis funkcije. Ovaj potprogram se koristi za slanje sekundarne adrese perifernom uređaju koji je priključen na serijski port. Pored pozivanja ovog potprograma, moramo pozvati funkciju TALK i u akumulator postaviti broj sekundarne adrese

Kako se koristi.

(1) Pozvati potprogram TALK
(2) U akumulator postaviti sekundarnu adresu
(3) Pozvati potprogram TKSA

UDTIM (C-64 i C 128)

Svrha Ažuriranje časovnika sistema
Pozivna adresa $FFEA (heksadekadna), 65514 (dekadna)
Korišćeni registri .A, X
Pripremni potprogrami. nema
Moguće greške nema

Opis funkcije Ovaj potprogram ažurira časovnik sistema Normalno se potprogram poziva pomoću KERNAL rutine prekida (IRQ) svakog 60 dela sekunde Ukoliko naš program obrađuje sopstvene prekide ovaj potprogram moramo pozvati da ažurira vreme

Kako se koristi

(1) Pozvati potprogram UDTIM

UNLSN (C-64 i C-128)

Svrha Slanje komande za prekid prijema
Pozivna adresa $FFAE (heksadekadna) 65454 (dekadna)
Korišćeni registri .A
Pripremni potprogrami. nema
Moguće greške Koristi READSS

Opis funkcije Potprogram koristimo kada želimo da pošaljemo komandu svim uređajima na serijskom kanalu — da prestanu sa prijemom podataka Obično ga koristimo kada smo završili sa komunikacijom između kompjutera i periferije koja je priključen na serijski kanal

Kako se koristi·

(1) Pozvati potprogram UNLSN

UNTLK (C-64 i C-128)

Svrha: Slanje komande za prekid predaje informacija
Pozivna adresa: $FFAB (heksadekadna) 65451 (dekadna)
Korišćeni registri: A
Pripremni potprogrami: nema
Moguće greške: Koristi READSS

Opis funkcije: Potprogram koristimo kada želimo da pošaljemo komandu svim uređajima na serijskom kanalu da prestanu sa slanjem podataka. Obično ga koristimo kada smo završili sa komunikacijom između kompjutera i periferije koja je priključena na serijski kanal.

Kako se koristi:

(1) Pozvati potprogram UNTLK.

VECTOR (C-64 i C-128)

Svrha: Postavljanje vrednosti vektora KERNAL-a
Pozivna adresa: $FF8D (heksadekadna) 65421 (dekadna)
Korišćeni registri: A, Y
Pripremni potprogrami: nema

Opis funkcije: Ovaj potprogram nam služi da iz dela RAM memorije, gde smo definisali vektore KERNAL-a, izvršimo izmenu KERNAL vektora. Pre pozivanja ove funkcije potrebno je da setujemo Carry flag i napunimo X i Y-registar adresom u RAM-memoriji gde smo definisali KERNAL vektore.

Kako se koristi:

(1) Setovati Carry flag.
(2) U X i Y-registar postaviti niži i viši bajt adrese gde su definisani vektori KERNAL-a.
(3) Pozvati potprogram VECTOR.

DODATAK „C"

REZOLUCIJA 640×200

U poglavlju o grafici prikazane su sve grafičke mogućnosti koje podržava Commodore-ov Basic 7.0. Kao što ste i sami mogli primetiti, grafičke mogućnosti su velike. Jedinu primedbu koju bismo stavili BASIC-u 7.0 je to što on ne podržava grafiku slike koju dobijamo preko RGB monitora.

U C-128 postoje dva čipa za generisanje slike. Jedan podržava sliku na televizoru ili kompozitnom monitoru, dok drugi čip podržava sliku na RGB monitoru. Zbog postojanja dva čipa za generisanje slike, na poleđini kompjutera se nalaze i dva izlaza ka monitoru. Ovo nije baš najsrećnije rešenje, jer za prelazak rada sa jednog čipa na drugi dovoljno je da pritisnete samo dva tastera (ESC, pa zatim X). Međutim, ako nemate ispred sebe dva monitora, onda morate imati dva kabla i stalno vaditi jedan i priključivati drugi. Preporučujemo vam da celom sistemu pridodate jedan naizmenični prekidač i time biste izbegli stalno vađenje i priključivanje kablova.

Rezolucija koja postoji na RGB izlazu je dvostruko bolja od rezolucija na televizoru ili kompozitnom monitoru. Dok na televizoru imamo maksimalnu rezoluciju od 320×200 tačaka, na RGB monitoru moguće je ostvariti rezoluciju od 640×200 tačaka. Na žalost, firma Commodore u BASIC 7.0 nije ubacila nijednu jedinu naredbu koja podržava rad u ovoj rezoluciji. Kad već imamo mogućnost bolje rezolucije šteta bi bilo ne iskoristiti je. Zbog toga vam dajemo mašinski program u obliku datoteke koji omogućuje rad u rezoluciji 640×200. Napominjemo još jednom da je ovu rezoluciju moguće ostvariti jedino na RGB monitoru ili televizoru koji ima RGB ulaz. Preko antenskog ulaza ne možemo dobiti sliku ove rezolucije.

Pored dvostruko veće rezolucije imamo i mogućnost rada u FAST modu (clock od 2Mh) što u osnovi povećava brzinu dva puta. To znači da pored veće rezolucije imamo i veću brzinu crtanja. Ove dve prednosti su dovoljan razlog da deo knjige posvetimo radu u ovoj rezoluciji.

Kada korektno ukucate priloženi program, startujte ga i snimite mašinski kod sa

```
BSAVE"HIRES-80",P4864 TO P6839
```

ako imate disk, a ako imate kasetofon uđite u MONITOR i iz njega snimite program na sledeći način

```
S HIRES-80 ,1,1300 1AB7
```

Nakon ovoga datoteka vam više nije potrebna. Pre početka rada učitajte program HIRES-80 i startujte sa SYSDEC("1303"). Nakon toga možete raditi u rezoluciji 640×200.

Ovaj mašinski program je pridodao samo jednu novu funkciju, i to GRAPHIC 6. Sve ostalo radite kao u normalnoj grafici visoke rezolucije. Na raspolaganju imate funkcije za kružnice, tačke, linije, pravougaonike... Da biste povećali brzinu crtanja, neka prva naredba vašeg programa bude FAST. Pošto u ovoj rezoluciji nemate mogućnost rada sa bojama, nemojte koristiti funkcije za boju. Isto tako, čip koji podržava ovu grafiku nema mogućnost rada sa sprajtovima.

Može se desiti da vam pri ulasku u mod visoke rezolucije desni deo ekrana treperi u širini jednog karaktera. Do toga je došlo zbog postojanja dve vrste čipova u C 128. Tu anomaliju jednostavno možete otkloniti ulaskom u MONITOR i promenom dva bajta u programu. Na adresama $1A60 i $1A85 postojeće vrednosti imaju poslednja tri setovana bita (jednaki su jedinici). Nedostatak ćete otkloniti ako ta tri bita dovedete na nulto stanje, tj. ako na te dve adrese upišete vrednosti #$40 i #$80 respektivno. Nakon ove ispravke snimite ponovo program i sve će biti u redu.

Rezoluciju od 640×200 koju dobijamo na RGB monitoru moguće je i promeniti. Može se rezolucija povećati po X-osi, a smanjiti po Y-osi, ili obrnuto (po želji). S obzirom na to da mašinski jezik nije tema ove knjige, programe koji menjaju rezoluciju možete pronaći u raznim časopisima koji se bave kompjuterima.

Pored programa u obliku datoteke imate i jedan Demo-program koji ilustruje rad u ovoj rezoluciji. Nakon izvršenog programa ostajete u visokoj rezoluciji, pa vam zbog toga dajemo jedan savet pri pisanju programa. Pored naredbe FAST za prelazak u brži režim rada, koristite i naredbu TRAP programska linija. Na datoj programskoj liniji napišite GRAPHIC 5:HELP. Ovo će vam omogućiti da se prilikom zaustavljanja programa normalno vratite u tekst-mod, a pored toga dobićete programsku liniju koja se izvršavala pri prekidu programa.

```
10 BANK 15:A=5475:G$=CHR$(7)+"GRESKA U BLOKU"
12 DO
14 FOR A=A TO A+3:POKE A,1:NEXT
16 FOR I=I+1 TO I+3
18 FOR A=A TO A+2:POKE A,I
20 NEXT:NEXT
22 LOOP UNTIL I>63
24 FOR I=0 TO 480:POKE 5746+I,PEEK(25000+I):NEXT
26 FOR I=0 TO 63:POKE 6227+I,PEEK(26519+I):NEXT
28 FOR I=0 TO 190:POKE 6291+I,PEEK(26254+I):NEXT
30 FOR I=0 TO 232:POKE 6482+1,PEEK(39728+I):NEXT
32 FOR I=1 TO 4
34 RESTORE I*100:S=0
36 READ P,K
38 FOR A=P TO K
40 READ D:POKE A,D:S=S+D
42 NEXT
44 READ D:IF D<>S THEN G=1:PRINT G$;I
46 NEXT:S=0
48 FOR I=1 TO 24
50 READ A:S=S+A
52 FOR A=A TO A+2
54 READ D:POKE A,D:S=S+D
56 NEXT:NEXT
58 READ D:IF D<>S THEN G=1:PRINT G$;5
70 IF G=0 THEN PRINT "NEMA GRESAKA'
99 REM *** BLOK 1 ***
100 DATA 4864,5110,32,153,19,76,6,19,120,169,45,14
1,8,3,169,19,141,9,3,88,169,0
101 DATA 141,0,255,169,6,141,6,213,169,32,133,48,1
33,50,133,52,169,0,133,47,133
102 DATA 49,133,51,96,32,128,3,201,222,144,31,201,
233,176,27,170,36,215,16
103 DATA 52,189,123,18,141,78,19,189,135,18,141,79
,19,138,32,128,3,32,0,0
104 DATA 76,246,74,32,134,3,76,243,74,62,114,215,1
29,147,141,43,86,88,226
105 DATA 121,96,26,22,103,23,24,101,100,24,105,105
,106,105,189,162,18,141
106 DATA 78,19,189,174,18,141,79,19,76,73,19,90,16
8,215,183,142,141,43,151
107 DATA 85,226,121,96,107,97,103,98,102,101,100,1
03,105,105,106,105,85,169
108 DATA 164,141,0,3,169,19,141,1,3,96,72,169,0,14
```

```
1,0,255,173,222,19,208,35
109 DATA 142,223,19,138,72,152,72,162,25,32,218,20
5,16,8,162,5,142,222,19
110 DATA 32,224,19,169,0,141,222,19,104,168,104,17
0,104,76,63,77,169,0,141
111 DATA 222,19,104,174,223,19,76,63,77,0,128,138,
72,169,0,141,0,255,162,25
112 DATA 32,218,205,16,3,76,93,26,104,170,72.76,11
0,107,24504
199 REM *** BLOK 2 ***
200 DATA 5120,5474,76,48,22,76,9,20,76,93,22,173,0
,255,72,169,0,141,0,255,169,7
201 DATA 141,6,213,104,141,0,255,96,173,0,255,133,
158,169,0,141,0,255,133,253
202 DATA 133,254,173,52,17,208,90,169,199,205,51,1
7,144,83,169,127,237,49
203 DATA 17,169,2,237,50,17,144,71,172,51,17,185,1
50,20,133,253,185,99,21
204 DATA 133,254,174,50,17,173,49,17,41,248,74,74,
74,24,125,147,20,133,252
205 DATA 24,165,253,101,252,133,253,165,254,105,0,
133,254,162,18,32,204,205
206 DATA 162,19,165,253,32,204,205,32,216,205,133,
252,173,49,17,41,7,170,189
207 DATA 139,20,24,96,56,96,128,64,32,16,8,4,2,1,0
,32,64,0,80,160,240,64,144
208 DATA 224,48,128,208,32,112,192,16,96,176,0,80,
160,240,64,144,224,48,128
209 DATA 208,32,112,192,16,96,176,0,80,160,240,64,
144,224,48,128,208,32,112
210 DATA 192,16,96,176,0,80,160,240,64,144,224,48,
128,208,32,112,192,16,96
211 DATA 176,0,80,160,240,64,144,224,48,128,208,32
,112,192,16,96,176,0,80
212 DATA 160,240,64,144,224,48,128,208,32,112,192,
16,96,176,0,80,160,240,64
213 DATA 144,224,48,128,208
214 DATA 32,112,192,16,96,176,0,80,160,240,64,144,
224,48,128,208,32,112,192
215 DATA 16,96,176,0,80,160,240,64,144,224.48,128,
208,32,112,192,16,96,176
216 DATA 0,80,160,240,64,144,224,48,128,208,32,112
,192,16,96,176,0,80,160
217 DATA 240,64,144,224,48,128,208,32,112,192,16,9
```

```
6,176,0,80,160,240,64,144
218 DATA 224,48,128,208,32,112,192,16,96,176,0,80,
160,240,64,144,224,48,128
219 DATA 208,32,112,192,41449
299 REM *** BLOK 3 ***
300 DATA 5680,5745,32,28,20,176,34,166,131,208,5,7
3,255,37,252,44,5,252
301 DATA 133,252,162,18,165,254,32,204,205,162,19,
165,253,32,204,205,162,31
302 DATA 165,252,32,204,205,165,158,141,0,255,96,3
2,28,20,176,245,37,252,240
303 DATA 6,32,87,22,162,0,96,32,87,22,162,255,96,8
138
399 REM *** BLOK 4 ***
400 DATA 6715,6838,76,0,20,201,158,208,11,32,34,16
0,32,128,3,169,0,133,216,96,32
401 DATA 244,135,224,6,240,41,176,36,76,224,19,234
,141,0,255,162,25,169,71,32
402 DATA 204,205,32,12,206,165,215,72,169,128,133,
215,32,66,193,104,133,215,104
403 DATA 72,170,76,110,107,76,40,125,169,0,141,0,2
55,162,25,169,135,32,204
404 DATA 205,32,28,158,169,0,141,0,255,224,2,176,2
29,224,0,240,29,160,64,132
405 DATA 8,160,0,152,162,18,32,204,205,162,19,32,2
04,205,162,31,32,204,205
406 DATA 136,208,250,198,8,208,246,96,14905
499 REM *** BLOK 5 ***
500 DATA 5845,32,0,20,6701,32,0,20,6285,32,29,26,6
599,32,29,26,6635,32,12,26
501 DATA 6663,32,12,26,6279,76,108,24,6288,76,108,
24,6065,32,82,25,6129,32,82,25
502 DATA 6276,32,82,25,6472,32,82,25,5778,32,6,20,
5830,32,6,20,5906,32,6,20
503 DATA 5959,32,6,20,5861,32,70,23,5882,32,70,23,
5952,76,181,22,6249,76,29,26
504 DATA 5746,32,50,158,6017,32,50,158,6291,32,50,
158,6227,234,234,234,151639

READY.
```

LITERATURA

[1] Compute!'s 128 Programer's Guide Greensboro North Carolina, 1985
[2] Compute!'s 128 System Guide Greensboro North Carolina, 1985
[3] Commodore 128, Bedienungshandbuch Commodore Buromaschinen GMBH Frankfurt 1985
[4] Commodore 64, Programer's Reference Guide. Commodore Busines Machines 1982
[5] Commodore 64, User Manual, Commodore Busines Machines 1982

Izdavač:
NIRO „Tehnička knjiga" Beograd, 7. jula 26

Za izdavača:
Sava Radović, direktor

Grafički urednik: Jugoslav Bogdanović

Naslovna strana: Slavko Milenković

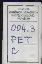
Korektor: Dušan Milanović

Štamparija: GRO „Kultura", OOUR „Slobodan Jović" Beograd, Stojana Protića 52

Tiraž: 2000 primeraka

Drugo izmenjeno izdanje 1988

Oslobođeno poreza na promet na osnovu mišljenja
Republičkog komiteta za kulturu SRS



YU ISBN 86-325-0104-6